8-1-2010

Thermal profiling of homogeneous multi-core processors using sensor mini-networks

Katherine Dellaquila

Thermal Profiling of Homogeneous Multi-Core Processors Using Sensor Mini-Networks

by

Katherine Ellen Dellaquila

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Engineering

Supervised by

Assistant Professor, Department of Computer Engineering Dr. Dhireesha Kudithipudi
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
August 2010

Approved By:

Dr. Dhireesha Kudithipudi
Assistant Professor, Department of Computer Engineering
Primary Adviser

Dr. Andres Kwasinski
Assistant Professor, Department of Computer Engineering

Dr. Roy Melton
Lecturer, Department of Computer Engineering

Thesis Release Permission Form

Rochester Institute of Technology

Kate Gleason College of Engineering

Title: Thermal Profiling of Homogeneous Multi-Core Processors Using Sensor Mini-Networks

I, Katherine Ellen Dellaquila, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or in part.

Katherine Ellen Dellaquila

Date

Dedication

This thesis is dedicated to my family, who have loved and supported me throughout all my endeavors.

Acknowledgments

I would like to thank my advisers, including Dr. Kwasinski and Dr. Melton and especially Dr. Kudithipudi, who dedicated her time to assist me even while on the other side of the globe.

I would like to thank the HotSpot team from the University of Virginia for sharing their modified version of SimpleScalar / Wattch, which was invaluable for my thesis validation.

I would also like to thank the RIT Department of Computer Engineering and all other faculty, staff, and classmates who helped me along the journey to getting my degree.

Abstract

With large-scale integration and high power density in current generation microprocessors, thermal management is becoming a critical component of system design. Specifically, accurate thermal monitoring using on-die sensors is vital for system reliability and recovery.

Achieving an accurate thermal profile of a system with an optimal number of sensors is integral for thermal management. This work focuses on a sensor placement mechanism and an on-chip sensor mini-network to combine temperatures from multiple sensors to determine the full thermal profile of a chip.

The sensor placement mechanism proposed in this work uses non-uniform subsampling of thermal maps with k-means clustering. Using this sensing technique with cubic interpolation, an 8-core architecture thermal map was successfully recovered with an average error improvement of 90% over sensor placement via basic k-means clustering. All the simulations were run using HotSpot 5.0 modeling Alpha 21364 processor as a baseline core.

The sensor mini-network using both differential encoding and distributed source coding was analyzed on a 1024-core architecture. Distributed source coding compression required fewer transmissions than differential encoding and reduced the number of transmitted bits by 36% over a sensor mini-network with no compression.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Large-scale integration and feature size reduction in transistors has led to increased power density and high temperatures in microprocessors. High temperatures in microprocessors introduce a number of magnified reliability weaknesses including more frequent timing errors [26], physical damage to the chip [8], and overall reduced circuit lifetime [67]. These effects have made thermal monitoring and management in microprocessors become an integral part of system design. Dynamic thermal management methods rely on accurate temperature data from on-die sensors and precise thermal monitoring analysis so that the appropriate actions can be taken to mitigate the high temperatures.

Ideally, a large number of sensors would be placed at a fine granularity over the entire chip to ensure adequate thermal coverage. Due to their additional area and power requirements, the quantity of sensors on a microprocessor must be limited [41]. Optimal locations for the few available sensors must be identified such that the thermal map of the chip can be reconstructed accurately. This process becomes an important task for accurate thermal monitoring.

Previous solutions that have attempted to monitor thermal activity via a uniform grid of sensors have resulted in unacceptable errors of up to $9.0°$C. Other proposed solutions employ various clustering algorithms to place sensors closest to those locations with significantly higher temperature, called *hot spots*. Hot spot clustering techniques, however, often result in placing many sensors only near hot spots and no sensors elsewhere. While these methods prove to be effective for detecting thermal emergencies, they seldom provide

insight into thermal activity across the entire chip.

In order to strike a balance between uniform temperature measurement and hot spot detection, this thesis incorporates non-uniform subsampling algorithms to select key thermal analysis locations on a chip. Clustering these sampled points places sensors for full thermal map coverage. Thermal map reconstruction through cubic interpolation from these sensors resulted in a 90% improvement in average error over sensor placement by way of basic k-means clustering.

As the number of cores in microprocessors increases, the quantity of thermal sensors in a single system will also escalate. In this thesis, the use of an on-chip thermal sensor mini-network (SMN) is explored to manage sensor data on this scale. Two different compression schemes for the SMN have been analyzed for many-core architectures to reduce bandwidth and power. Applying differential encoding reduced network traffic, yet introduced sensor-to-sensor communication. SMN compression through distributed source coding showed to be the best compression scheme due to no communication between sensors. This scheme was able to reduce the number of transmitted bits by 36% in the presented example of a 1024-core architecture.

This thesis document is organized as follows: Chapter 2 discusses further motivation behind thermal monitoring in microprocessors. Previous thermal monitoring mechanisms and sensor placement through non-uniform subsampling are discussed in Chapter 3. The simulation framework and test data used in analysis are explained in Chapter 4. Chapter 5 gives an analysis and discussion of thermal sensor placement mechanisms for an 8-core architecture. The sensor mini-network and proposed compression schemes are explained in Chapter 6. Concluding remarks and proposed future work are discussed in Chapter 7.

Chapter 2

Motivation

Today's ever increasing computational demands for higher operating frequencies and smaller devices has driven designers of modern processors to take advantage of large-scale integration and feature size reduction in transistors. Aggressive technology scaling and higher integration density, however, introduce additional design challenges to account for the resultant decrease in circuit reliability. Circuits become less reliable in terms of more frequent timing failures, hindered speed, and reduced circuit lifetime. High performance VLSI circuits sustain such reliability consequences due to greater process variation [43], higher current densities in interconnects [67], and increased power density on microprocessors [59]. These weaknesses are anticipated to become much more significant in exascale computing [32].

Figure 2.1 shows the past power densities for single-core chips and multi-core chips as well as the ITRS projections through 2020 [59]. Power density has increased significantly through the 1990s as a result of decreasing feature size. A maximum density of approximately 100 watts per square centimeter was reached, and held at this magnitude via a reduced clock rate. Further clock rate degradation is no longer an option to limit power density and simultaneously maintain performance. The number of cores in a processor must be significantly increased to maintain and improve performance, leading into the exascale computing realm. The ITRS projections in Figure 2.1 for power density in upcoming years repeatedly increase until the maximum density on the order of 100 watts per square centimeter has been reached. The power density is reduced again due to improvement in

low power processors, though it is not projected to drop to an insignificant level.

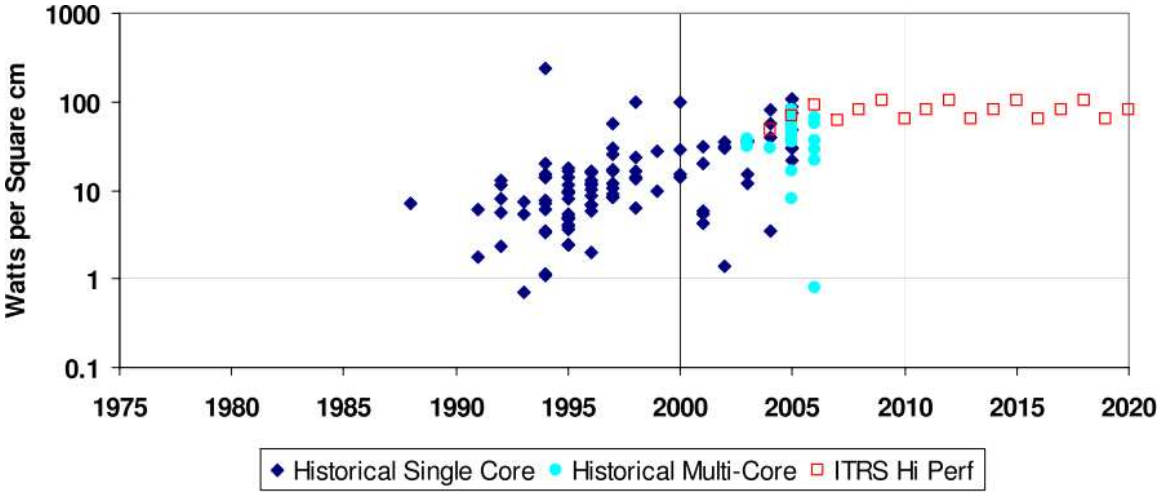


Figure 2.1: Power density history and projections from [31] and [59].

This high power density trend has led to overall elevated on-chip temperatures and localized areas of significantly higher temperatures, referred to as hotspots. It has been reported in [19] that the thermal gradient across chips has reached as high as $50°$C, and this value rises with higher operating frequencies. High temperatures and large thermal variation across the chip introduce an array of reliability hazards that risk both unexpected circuit functionality and weakened physical parameters of the chip. The work in this thesis aims to help avoid the occurrence of thermal repercussions on a chip by monitoring the across-chip temperatures at run-time.

Timing errors become more frequent with increased on-chip temperature. It has been reported in [26] that if the temperature of a circuit is elevated by $10°$C, Elmore (interconnect) delay will increase by 5%. Extended delay in the interconnects is capable of producing timing errors that will require additional delays from which to recover.

Previous work in [1] and [26] has shown that as the temperature of a circuit increases, interconnects are weakened. Leakage current grows exponentially due to the electromigration (EM) phenomenon [1]. This phenomenon describes how high temperatures allow electrons to migrate, resulting in a weaker, thinner conductor at the interconnects. Modern-day

processors with dimensions under 0.18 microns are inherently protected against electromigration at normal temperatures, but start to become vulnerable at approximately 75-85°C and higher temperatures [33]. As further emphasized in [68], thermal effects become even more significant at dimensions under 0.1 microns due to this phenomenon.

Physically, the overall lifetime of the chip is very likely to be reduced with a rise in temperature. As shown in [67], the mean time to failure (MTTF) of the circuit decreases exponentially with increase in temperature. The chip/package interface material is likely to crack under thermal stress [8].

To avoid these negative effects from large temperature gradients, the on-chip temperatures must be monitored and controlled at run-time. On-die temperature sensors or Negative Bias Temperature Instability (NBTI) sensors can be used to measure temperatures continually at certain locations on a chip so that thermal data can be analyzed. There are several dynamic thermal management (DTM) methods being used in modern processors that take appropriate action based on thermal sensor readings and proper assessment. Intel Pentium 4, Pentium M, and IBM's PowerPC contain physical thermal sensors that trigger alerts should they encounter a temperature reading above a specified threshold [52][55]. Clock throttling is used in these processors to regulate power consumption and lower the chip's temperature. Intel's Centrino Processor uses an implementation of dynamic voltage scaling (DVS) to manage temperature [28][9].

DTM mechanisms rely on accurate and precise thermal maps at run-time across the chip. An inaccurate thermal map reconstruction showing temperatures running higher than the actual on-chip temperatures could trigger a thermal management scheme to take action for high temperatures when it is not necessary. This needlessly adds delay and increases power consumption. On the contrary, an inaccurate thermal map giving temperatures lower than the actual thermal data could prevent a thermal management scheme from taking the appropriate actions when they are needed. This scenario could lead to any of the negative effects of high temperatures discussed previously.

Under ideal circumstances, many sensors would be placed at a fine granularity over

an entire chip to ensure adequate coverage of all thermal events. Such a design, however, is not practical due to the fact that a large number of sensors will increase costs in terms of area, power, and routing complexity [41]. To obtain the most accurate thermal map of the whole chip without utilizing a large number of sensors, a methodical sensor-placement optimization scheme must be derived that will allow for the gathering of sufficient thermal data.

Thermal management techniques rely on measurements returned by sensors that have been placed at optimal locations throughout the chip. Typically, the data returned from a sensor is not compared and analyzed together with the temperature data from other nearby sensors. Each sensor's reading is assumed to be representative of the region surrounding it without consideration of temperatures measured by other sensors. The work in this thesis takes advantage of using an on-chip thermal sensor mini-network to monitor the thermal events on a chip sufficiently and efficiently.

Chapter 3

Thermal Sensor Placement Mechanisms

Several research groups have developed optimization algorithms that result in using a minimum number of sensors while maintaining adequate coverage. These optimization algorithms fall into two main categories: uniform and non-uniform sensor placement.

3.1 Uniform Sensor Placement

Uniform sensor placement optimization schemes are intended for use with chips that have an unknown typical thermal pattern. The sensors are placed in a uniform static grid throughout the entire chip with the intention of being able to detect all temperature violations, regardless of where they occur on the chip. As mentioned previously, only a finely-grained grid of sensors is capable of achieving near-perfect accuracy. Due to significant cost restrictions associated with sensor overheads, the granularity of the grid must be bound, limiting the accuracy of this model.

A straight-forward linear interpolation approach is proposed in [44] to account for this restriction and refine the temperature measurements. Considering the sensors displayed in Figure 3.1, it is assumed that sensor T_4 has measured the highest temperature. From this information, it can be deduced that the hottest point in this region is located within the dashed square in the figure. The edges of the square are located exactly midway between T_4 and the neighboring sensors. To refine further the location of the hot spot, the temperatures of the neighboring sensors are compared: T_3 and T_5 to refine in the x dimension and T_1 and

T_7 to refine in the y dimension. The interpolation scheme with a 4×4 grid of sensors was shown in [44] to improve upon a static uniform grid of the same size with no interpolation by an average of $1.59°$C across the SPEC2000 benchmarks [66] in a single-core processor.

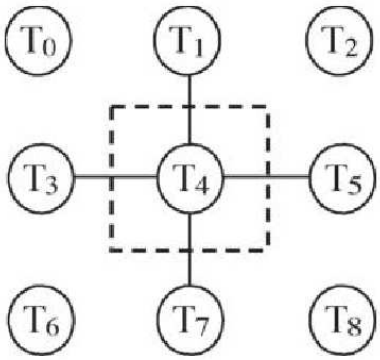


Figure 3.1: Uniformly placed sensors with interpolation used in [44].

A slight advantage of implementing a uniform thermal sensor allocation technique is that it does not rely on thermal profiling data. No knowledge of hot spot locations and temperatures needs to be acquired prior to implementing a technique of this type. This characteristic does, however, limit the accuracy of the uniform grid model because the distances between the sensor locations and the hot spots cannot be minimized. Without knowledge of common resulting thermal maps, sensors arranged in a uniform grid will not aways be able to detect hot spots as accurately as the same number of sensors located near common hot spots.

3.2 Non-Uniform Sensor Placement

Non-uniform sensor placement optimization schemes are intended for use where thermal maps from typical chip execution across several applications are available for analysis. These types of techniques take advantage of the known hot spots on the chip to determine the most advantageous locations for sensors to be placed. A naive approach is to place a

sensor on each hot spot found through thermal profiling across several applications. Unfortunately, this approach is not practical because a high number of hot spots is very likely to result, and using a large number of sensors is not practical. Ideally, a minimum number of sensors would be arranged on the chip such to provide coverage of all possible hot spots. It has been shown in [63] that hot spots will not always remain in the same locations on the chip during execution of a single program, and various applications running on the same chip will show hot spots in different regions. Hot spot locations and temperatures are application dependent, and it is not likely that a solution optimized for a single application will be sufficient for the others. One sensor placement configuration must suffice for all hot spots that may arise during the execution of any program.

Several methods that detect thermal violations with a limited number of sensors have been developed based on hot spot locations and temperatures found via thermal profiling. Skadron et al [62] have proposed Equation 3.1 to describe the maximum radius R between a hot spot and a potential thermal sensor location, while capping the error to a degree ΔT. The value T_{max} denotes the difference between the maximum and minimum temperature value in the chip. In this equation, K is used to represent the effects of the materials of which the chip is made. This includes the thickness of the processor package-die, heat spreader, and thermal interface material multiplied by thermal resistivity factors specific to each material.

$$R = 0.5 \cdot K \cdot \ln(\frac{T_{max}}{T_{max} - \Delta T}) \tag{3.1}$$

3.2.1 Quality-Threshold Clustering

The algorithm described in [69] incorporates Equation 3.1 with the quality threshold (QT) clustering algorithm commonly used in gene clustering [11]. Treating the hot spots as points that must be clustered, the hot spot groupings and corresponding sensor locations are determined based on the values of T_{max} for all of the hot spots in each respective cluster. QT clustering is an iterative technique that assigns hot spots to clusters based on

their physical locations on the chip relative to the other hot spots. The sensor location for each cluster is refined after the addition of a candidate hot spot to be the centroid of the included hot spots, thus obtaining the best possible sensor location for the given set of hot spot data points. The newly added hot spot will be kept in this cluster only if every other hot spot in the cluster is located within the distance R from the cluster center.

The work in [69] resulted in placing 23 sensors with $T_{max} = 3°$C using the QT clustering method on an Alpha 21364 processor core and hot spots produced by the SPEC2000 benchmarks. The 23 sensors sensed the complete thermal profile of the core with an average error of $0.2899°$C.

Though this algorithm proves to be sufficient for monitoring thermal events, it does not incorporate the number of clusters or sensors that are available to use for a specific design. This could be detrimental for several reasons. The algorithm does not end execution until every hot spot is placed in a cluster, creating new clusters where necessary to include hot spots that are located far away from the others. The number of sensors required by the QT clustering algorithm may not be available for use in a practical design. To place fewer sensors using QT clustering, the allotted hot spot-sensor distance maximum must be increased, which may decrease the accuracy of the entire model's results rather than only for the outlying hot spots.

3.2.2 K-means Clustering

The more popular basic k-means clustering algorithm requires the number of sensors to be placed as an input parameter, k. The hot spots are placed into k different clusters, with a temperature sensor placed at the centroid of each cluster. The cluster assignments are chosen such that the mean squared distance from each hot spot to the nearest cluster center is minimized [42]. First, the k cluster centers are chosen randomly from the set of known hot spot points. Each hot spot is then assigned to a cluster C_j such that Euclidean distance $E(O_j, h_i)$ between the hot spot h_i and this cluster's center O_h is minimized. The equation to determine the Euclidean distance between two points is shown in Equation 3.2. In this

equation, (h_{ix}, h_{iy}) represents the location of a hot spot h_i and (O_{jx}, O_{jy}) represents the location of a cluster center O_j in the (x,y) plane.

$$E(O_j, h_i) = (O_{jx} - h_{ix})^2 + (O_{jy} - h_{iy})^2 \tag{3.2}$$

At the end of each iteration, each cluster center is updated to be the centroid of the locations of all hot spots assigned to that cluster. The Euclidean distances between the hot spots and the cluster centers are then recomputed. If a new minimum distance between a hot spot and a different cluster center is found, the hot spot is reassigned to the corresponding cluster. This process is repeated until no hot spot has been reassigned to a different cluster or the total sum of all Euclidean distances does not have a significant increase.

A thermal gradient-aware version of the k-means clustering algorithm has been proposed in [45]. The main goal of this approach is to place the temperature sensors to those hot spots that typically have higher temperatures. The clusters are formed in 3-D space using each hot spot's temperature, t, as the third dimension of calculating Euclidean distance, as shown in Equation 3.3.

$$E(O_j, h_i) = (O_{jx} - h_{ix})^2 + (O_{jy} - h_{iy})^2 + (O_{jt} - h_{it})^2 \tag{3.3}$$

The cluster centers are updated in each iteration with consideration of hot spot temperature. The hot spots are weighted in the centroid calculation relative to the magnitude of their temperatures. As in the basic k-means clustering algorithm, the cluster centers and hot spot cluster assignments are iteratively refined until no hot spot has been reassigned to a different cluster or the total sum of all 3-D Euclidean distances does not have a significant increase.

As shown in [45], the thermal gradient-aware k-means clustering resulted in an average error at best of $2.10°$C placing 16 sensors over a single core and an average error of $1.63°$C using 36 sensors. Using the same two numbers of sensors with identical thermal profiling data, the basic k-means clustering algorithm resulted in an average error of $4.58°$C and

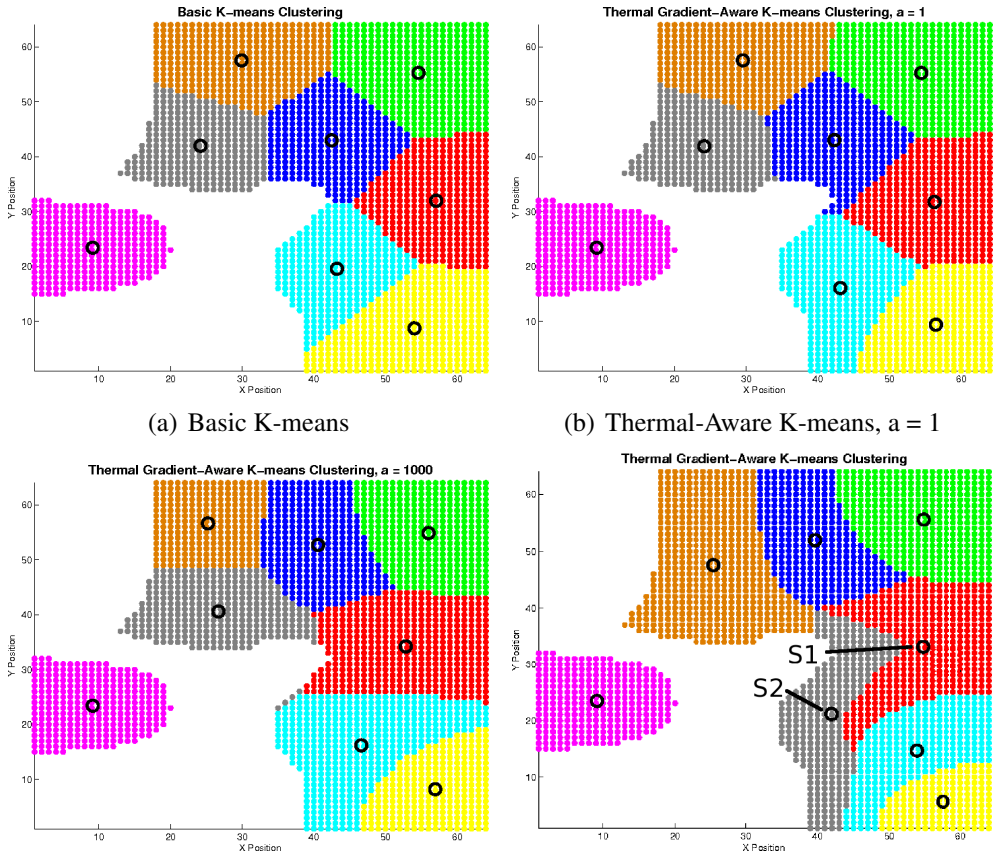
$3.05°$C. This shows $2.48°$C and $1.42°$C improvements, respectively. All experiments were run using HotSpot configured for the Alpha 21364, and hot spot positions were determined from thermal patterns pertaining to the SPEC2000 benchmarks [66].

Although thermal gradient-aware k-means clustering works well under many conditions, this technique is not always optimal in complex hot spot distribution scenarios and may produce solutions worse than the basic k-means approach. Hot spots are often sorted into inappropriate clusters due to their common temperature and regardless of their physical locations on the chip. Figures 3.2(a) and 3.2(b) show k-means clustering results using basic and thermal-gradient aware on the same set of hot spot points with eight sensors. Although the clustering of hot spots varies slightly for the two methods, the sensor placement locations are almost identical. To make a difference in sensor placement, the temperatures of the hot spots used in thermal gradient-aware k-means calculations can be scaled according to Equation 3.4, where a is a constant specifying the steepness of the temperature gradient.

$$New\,Temperatures = a \cdot \frac{Original\,Temperatures}{Maximum\,Temperature} \tag{3.4}$$

Figures 3.2(c) and 3.2(d) show the clustering results of using Equation 3.4 with $a = 1000$ and $a = 2500$, respectively, on the same hot spot set used in Figures 3.2(a) and 3.2(b). In both situations, the sensors have been placed closer to the hot spots of higher temperature and further from the hot spots of lower temperature. Many of the hot spot cluster assignments, however, are not appropriate spatially. In Figure 3.2(d), for example, many of the red hot spots clustered with sensor S1 would be more appropriately grouped with the gray hot spots clustered with sensor S2, and vice versa.

The work in [41] shows that while thermal-gradient aware k-means clustering is effective for single-core processors, it is not appropriate for multi-core processors with strong inter-core thermal interaction due to the unlikelihood of hot spots appearing in the same locations on every core.



(a) Basic K-means

(b) Thermal-Aware K-means, a = 1

(c) Thermal-Aware K-means, a = 1000

(d) Thermal-Aware K-means, a = 2500

Figure 3.2: K-means clustering sensor placement variations.

3.2.3 Hot Spot Determination in Regard to Sensor Allocation

There are many properties to consider when determining which locations on the chip are considered to be ”hot spots.” Varying the determination rules has the potential to significantly affect the resulting sensor placement locations. The trade-offs between full thermal map characterization and hot spot detection must be considered when identifying initial hot spot locations.

Local Hot Spots

One common determination rule is to assign a specified number of hotspots per functional block within the processing core, referred to as *local hot spots* [44][69]. This technique

encourages sensor placement across the entire die and is most appropriate for characterizing the full thermal map of the processor. The hot spot locations will be the points that reach the local maximum temperature for each functional block. For many functional blocks, the hottest points will be the near the edges of the block adjacent to a hotter component.

Global Hot Spots

A second method of hot spot determination is to record *global hot spots*, or any location on the die that reaches or surpasses a specified emergency temperature threshold, typically near $82°$C or 355 K [41][44]. Temperature sensors will be placed closer to the locations on the chip of significantly higher temperature, and thus will not likely be spread across the chip. This technique is best for quickly recognizing emergency temperatures as opposed to recovering the full thermal map of the entire processor. Furthermore, the number of hot spots determined by this technique is inversely correlated with the specified emergency temperature threshold. A higher threshold will result in fewer hot spots that could all be located within a single functional block in the processor. Alternatively, a low enough threshold will result in many hot spots, which could potentially cover more than 50% of the processor. A large number of hot spots would allow sensors to be spread through a larger area. As reported in [64], the integer register file is repeatedly the hottest component in the Alpha 21364 core across the SPEC2000 benchmarks. Choosing a high emergency temperature threshold could result in placing sensors only in the integer register file, while a lower threshold would allow sensors to be placed in adjacent functional blocks across the processor. The tradeoffs between quick hot spot detection and full thermal map recovery must be considered.

3.3 Non-Uniform Subsampling of Thermal Maps

In many-core architectures, there is a high likelihood of measuring a very large number of global hot spots. The number of hot spots may be so large such that clustering algorithms

are not very effective in placing sensors near the hottest points. To reduce the number of points to be clustered while maintaining clear representation of thermal data, non-uniform subsampling algorithms can be used obtain a subset of key thermal analysis locations on a chip. More samples are selected from regions of higher temperature and fewer points are selected from regions of lower temperature. Subsampling a thermal map will likely strike a balance between uniform temperature measurement and thermal emergency detection. After the thermal map has been subsampled, clustering algorithms can be used on the subsamples to determine sensor placement locations.

The two gradient based non-uniform subsampling algorithms for images proposed in [54] select sample pixels from a given image such that a constant gradient region will be represented by a number of samples linearly proportional to the gradient magnitude.

Deterministic Subsampling

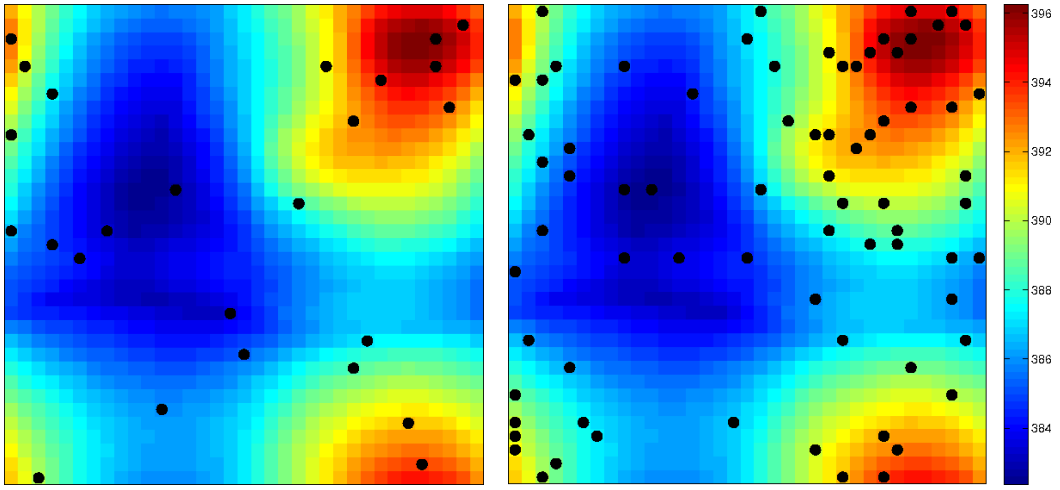
The deterministic version of this algorithm states that in order to quantize the data set $||\nabla I_1||$ into Q levels, a list of pixel locations I_q with a quantized gradient norm of q must be built for each level q. After all pixels have been distributed into appropriate quantization levels,every s_qth pixel in each list I_q will be selected, where $s_q = \lceil c/q \rceil$ for a constant c. This specification ensures that samples are selected more frequently in regions of high gradient. The constant value c is adjusted to yield a larger or smaller number of samples.

Applying subsampling algorithms to a full thermal map of a processor core while treating the temperature values as gradients results in more samples closer to the regions with more hot spots and fewer samples near cooler regions. Adjusting the value of c affects the number of samples taken from a thermal map. Before performing subsampling, the temperature should be normalized according to Equation 3.5 using the minimum temperature T_{min} and maximum temperature T_{max} in the thermal map used for analysis.

$$I_{norm} = \frac{||\nabla I_1|| - T_{min}}{T_{max} - T_{min}} \tag{3.5}$$

Performing the deterministic non-uniform subsampling algorithm on a sample thermal

map yielded the sampled results displayed in Figure 3.3(a) and 3.3(b). Both runs used 25 quantization levels. Figure 3.3(a) shows the results from setting the constant $c = 5$. This constant produced many fewer samples than setting the constant $c = 0.25$ as shown in Figure 3.3(b). Both plots reveal sampling locations spread throughout the entire thermal map. This algorithm is fairly conservative and similar to uniform sampling.



(a) Thermal deterministic subsamples from $c = 5$

(b) Thermal deterministic subsamples from $c = 0.25$

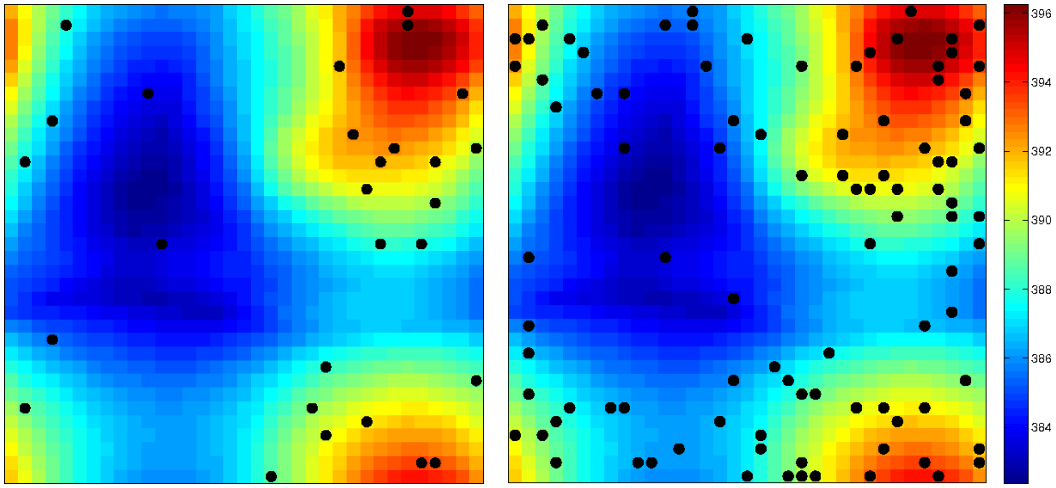
Figure 3.3: Thermal deterministic non-uniform subsampling results with 25 quantization levels.

Stochastic Subsampling

A stochastic version of non-uniform sampling looks at each individual pixel location (i, j) and decides whether to select this pixel as a sample or not. Pixels are selected with probability $p(i, j) = min(\alpha * ||\nabla I_1||(i, j), 1)$. Adjusting the proportionality constant α yields fewer or additional samples.

Performing the stochastic non-uniform subsampling algorithm on a sample thermal map yielded the sampled results displayed in Figure 3.4. Figure 3.4(a) shows the results from setting the constant $\alpha = 1.5$, which produced fewer samples than setting the constant $\alpha = 5$ as shown in Figure 3.4(b). Both plots show many sampling points in the hottest

regions, and only one or two samples in the coolest region. The samples taken in this stochastic subsampling algorithm accurately reflect the thermal gradient of the chip.



(a) Thermal stochastic subsamples from $\alpha = 1.5$

(b) Thermal stochastic subsamples from $\alpha = 5$

Figure 3.4: Thermal stochastic non-uniform subsampling results.

K-means clustering can be used to determine appropriate sensor locations when treating the samples as points to be clustered. Due to the sampled locations, the resulting thermal sensor placement will be able to maintain a balance between profiling the entire core and detecting thermal emergencies.

Chapter 4

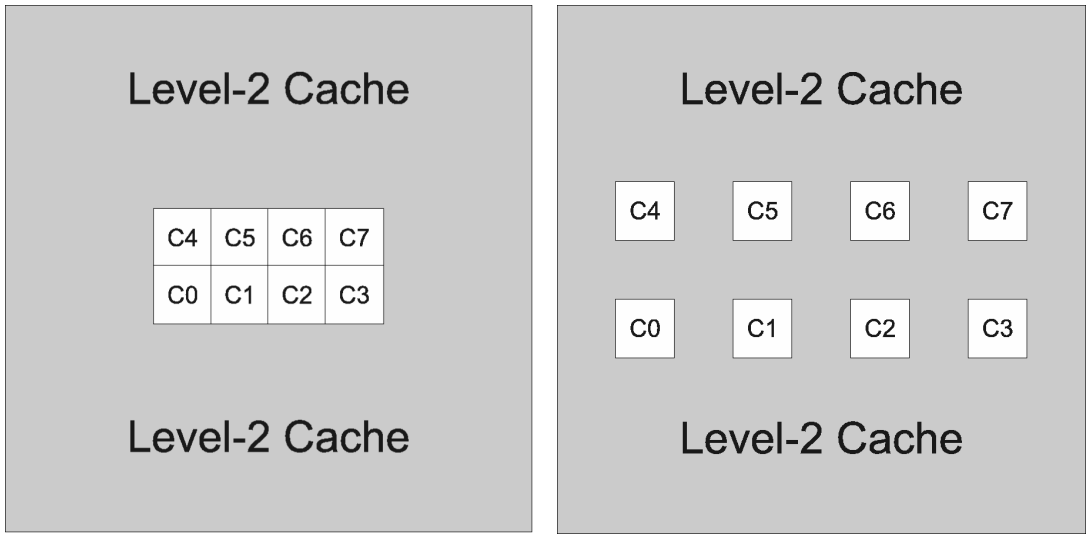
Simulation Framework

4.1 Multi-Core Architectures

Due to their improved performance per watt efficiency [50], multi-core processors have become the new industry standard, packing more cores onto a single die with each generation [5][31]. Positioning of the cores within a processor floorplan affects the thermal distribution and maximum temperatures. Two common layouts are displayed in Figure 4.1 for 8-core architectures. Recent trends indicate that the dense floorplan with cores placed immediately next to each other in Figure 4.1(a) is more popular in modern-day many-core processors [5] [50]. The sparse floorplan in Figure 4.1(b) has been used to prevent thermal coupling between the cores [36]. Both floorplan options were analyzed in this thesis.

4.2 Simulation Platform

The Alpha 21364 processor core was used in these experiments as a baseline core [47]. The 21364 core has been used repeatedly for thermal analysis and thermal management research [14][41][45]. Table 4.1 shows parameters for the 21364 core. The architecture of the 21364 core is shown in Figure 4.2. To accurately use the 21364 processing core as each core in a multi-core architecture, the floorplan must be scaled down to account for the decrease in transistor feature size. The original 21364 was implemented in 130 nm technology and is scaled to 45 nm for the following simulations.



(a) Dense 8-core architecture.

(b) Sparse 8-core architecture.

Figure 4.1: 8-Core architecture floorplans used in simulation.

Alpha 21364 Parameters	
Level 1 instruction and data caches	4-way associative 64 KB with 32-byte block size 2 cycle latency.
Level 2 cache	Unified 4-way associative 512 KB with 128-byte line size 15 cycle latency
Load store queue	32
Register update unit	64
Nominal Frequency	3 GHz
Nominal V_{dd}	1.3 V

Table 4.1: Alpha 21364 parameters [47]

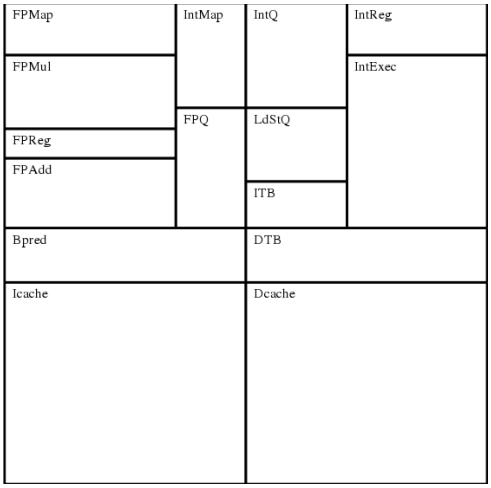


Figure 4.2: Alpha 21364 core architecture.

The SPEC2000 benchmark suite [66] was simulated using SimpleScalar Version 2.0 [11]. SimpleScalar is a well-known cycle-by-cycle functional microarchitectural simulator. The version of SimpleScalar used in this thesis was configured to more accurately model the inner workings of the Alpha 21364 processor. A microarchitectural-level power analysis tool, Wattch [10], was integrated with SimpleScalar to obtain realistic dynamic power trace data every 10,000 cycles for the 21364. Leakage power was modeled as a percentage of each functional block's dynamic power. Additional leakage power resulting from high temperatures was provided in thermal simulations.

The thermal simulations were conducted using HotSpot 5.0, a widely accepted microarchitectural thermal simulator [65]. The dynamic power trace data from Wattch was scaled appropriately and then input to HotSpot for each of the multi-core architectures. The default parameters for HotSpot were used with the exception of the spreader and sink parameters, which must be modified to accommodate the larger architectures.

4.3 Modifications to the HotSpot Framework

To ensure that power trace data from Wattch has been scaled appropriately for the chosen technology sizes in these experiments, Equation 4.1 was used in conjunction with technology trend data from the International Technology Roadmap for Semiconductors [59] and Dennard scaling [17]. The equation for dynamic power, $P_{dynamic}$, reveals that average device capacitance and operating frequency scale linearly with power, while V_{dd} is squared.

$$P_{dynamic} = 0.5\alpha C V_{dd}^2 f \tag{4.1}$$

Operating frequency has not scaled linearly with technology. ITRS reports a 1.73 factor increase in operating frequency from 3 GHz in the 21364 to 5.2 GHz in 2005 90 nm technology. A further increase by a factor of 1.128 to 5.87 GHz in 45 nm 2010 technology was reported by ITRS. Together, these factors bring a total of a 1.95 scaling factor from 130 nm technology. Dennard scaling reports that average device capacitance has scaled linearly by factor of 0.7. Recent ITRS trends show that this linear increase has slowed to a factor of 0.55 since 2005. Combining these factors gives a total capacitance scaling factor of 0.3885. The 130 nm 21364 was implemented with $V_{dd} = 1.3$V. ITRS reported that at 45 nm technology in 2010, $V_{dd} = 0.97$V. This gives a V_{dd} scaling factor of 0.769^2. Combining all scaling factors together gives a total dynamic power scaling factor from 130 nm to 45 nm of 0.38.

To ensure that a realistic heat sink and spreader were modeled appropriately for a scaled architecture, both the physical size and the average expected power P_{avg} of each die were taken into account. The chosen spreader size was set to twice the size of the die and the sink was set to four times the size of the die. These sizing ratios were chosen in accordance with similar research done in [41] and to mimic various modern processors [27][53]. Heat sink convection resistance was calculated according to Equation 4.2 using an average temperature T_{avg} of 333 *K* and ambient temperature T_{amb} of 318 *K*. This equation ensures that the heat sink will have enough capacity to transfer the expected heat resulting from the

specified average power. Using these specifications, the sink and spreader parameters were chosen as shown in Table 4.2 for the architectures used in the following experiments.

$$Convection\ resistance = \frac{T_{avg} - T_{amb}}{P_{avg}} \tag{4.2}$$

Component	Scaled Size
Die size	0.015 m
Spreader size	0.031 m
Sink size	0.062 m
Convection resistance	0.1645 (K/W)

Table 4.2: Sink and spreader sizes used in HotSpot simulation for the 8-core architectures.

Additional HotSpot configuration modifications included the thickness of the sink, spreader, die, and thermal interface material. These parameters were set to the same values presented in [41] and are displayed in Table 4.3. All other parameters were left as the default HotSpot values. To reflect realistic sensing capabilities, all sensors are assumed to be accurate within 2 *K* of the true thermal data.

Parameter	Value
Spreader thickness	0.1 cm
Sink thickness	0.7 cm
Die thickness	0.05 cm
Thermal interface material thickness	0.0075 cm

Table 4.3: Default HotSpot configuration modifications.

HotSpot 5.0 was run in grid mode to more accurately obtain temperatures across the entire die. In grid mode, HotSpot divides the entire floorplan into a grid and calculates the temperature for each grid cell individually. A finer grid size gives a higher resolution and therefore a more accurate estimate of realistic across-chip temperatures. A finer grid size, however, also increases simulation time and thus must be limited. A grid size of 256 x 256 for both 8-core architectures was chosen to obtain accurate spatial temperature data without requiring significant computation time.

4.4 Representative Benchmarks

The SPEC2000 benchmarks were used in this thesis. The work in [64] addresses the power and thermal characteristics of the SPEC2000 benchmarks on the Alpha 21364. A representative set of 11 of the 26 available benchmarks were presented. A summary of the chosen benchmarks is displayed in Table 4.4. Each of these benchmarks was run through an integrated version of SimpleScalar/Wattch to obtain detailed power trace data for the single core 21364. Each benchmark was fast-forwarded to a representative sample of 500 million instructions and run through HotSpot twice. The first run was used to represent warm-up of all components and input as initial temperature for the second run. Temperature results from the second run were used in all further analysis.

	IPC	Average Power (W)	% Cycles in Thermal Violation	Dynamic Max Temp (°C)	Steady State Temp (°C)	Sink Temp (°C)
Low Thermal Stress (cold)						
parser(I)	1.8	27.2	0.0	79.0	77.8	66.8
facerec(F)	2.5	29.0	0.0	80.6	79.0	68.3
Severe Thermal Stress (medium)						
mesa(F)	2.7	31.5	40.6	83.4	82.6	70.3
perlbmk(I)	2.3	30.4	31.1	83.5	81.6	69.4
gzip(I)	2.3	31.0	66.9	84.0	83.1	69.8
bzip2(I)	2.3	31.7	67.1	86.3	83.3	70.4
Extreme Thermal Stress (hot)						
eon(I)	2.3	33.2	100.0	84.1	84.0	71.6
crafty(I)	2.5	31.8	100.0	84.1	84.1	70.5
vortex(I)	2.6	32.1	100.0	84.5	84.4	70.8
gcc(I)	2.2	32.2	100.0	85.5	84.5	70.8
art(F)	2.4	38.1	100.0	87.3	87.1	75.5

Table 4.4: Thermal information for the SPEC2000 benchmarks, generated in [64].

The representative set of benchmarks are classified into categories of Low Thermal Stress, Severe Thermal Stress, and Extreme Thermal Stress based on the proportion of simulated cycles experiencing a thermal violation at any location on the chip. Thermal violation for these simulations was defined as observation of a temperature greater than or

equal to 82°C. For the simulations in this thesis, benchmarks were chosen based on the listed thermal characteristics and thermal stress categories.

To simulate multi-core processors, power trace data specific to each benchmark was replicated, scaled appropriately, and assigned to individual cores in the multi-core configurations in various combinations. As done in [64], each benchmark was fast-forwarded to a representative sample of 500 million instructions and run through HotSpot twice, using the first run to represent warm-up and as an input as initial temperature for the second run. Temperature results from the second run were used in all further analysis in this thesis. The benchmarks chosen for simulation of the 8-core architectures are displayed in Table 4.5, where each core represents those labeled previously in Figures 4.1(a) and 4.1(b). The first set contains benchmarks all from the Extreme Thermal Stress category, and the second contains a mix of benchmarks from all three categories. These two test sets offer a variety of interesting thermal patterns for analysis.

Core	**Set 1: Hot Benchmarks**	**Set 2: Mix of Benchmarks**
C0	art	parser
C1	gcc	art
C2	vortex	bzip2
C3	eon	bzip2
C4	eon	parser
C5	vortex	art
C6	gcc	gcc
C7	art	bzip2

Table 4.5: Benchmark assignments for test sets in the 8-core architectures.

Chapter 5

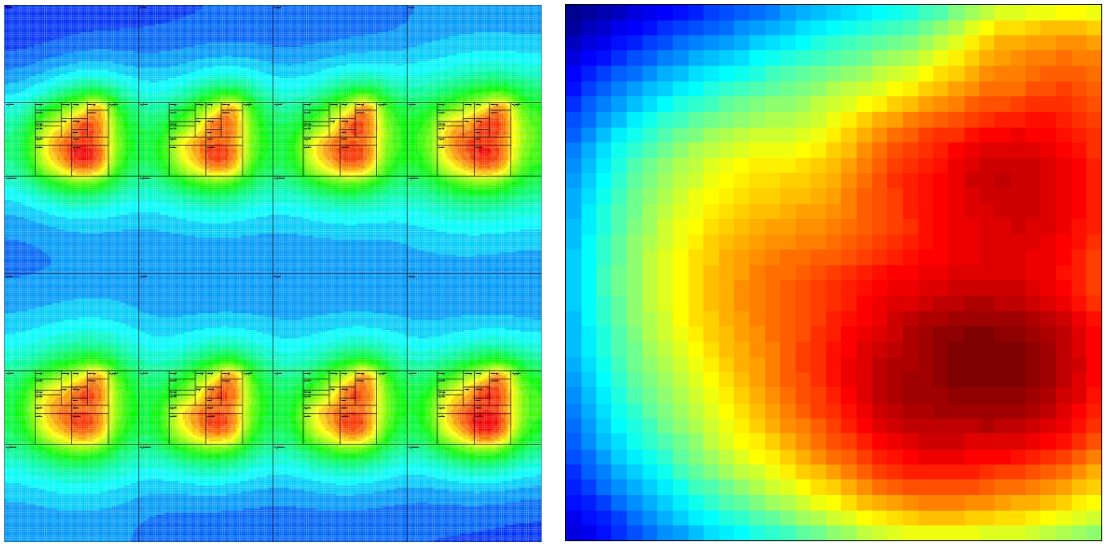
Analysis of Thermal Sensor Placement Mechanisms

To determine sensor placement, the benchmark with the highest average power and hottest average temperatures, *art*, was applied to every core in each architecture. HotSpot simulations were conducted with this benchmark configuration to determine the common hot spot locations. The hottest temperatures encountered for every location on the chip throughout the simulation were recorded. Maximum temperatures for all eight cores were folded onto a single core to obtain a thermal map of all maximum temperatures seen on a core. A thermal gradient map of the resulting temperatures is shown in Figure 5. The hottest functional block for the sparse architecture was the data cache.

Hot spot locations were determined from this thermal map. For the standard k-means clustering simulations used in this thesis, any location on the chip that has recorded a temperature of 82°C or higher at any time is considered a hot spot, while the non-uniform subsampling simulations analyzed the entire thermal map before clustering.

The basic k-means clustering algorithm was implemented on the folded maximum temperatures core from the 8-core sparse architecture for eight thermal sensors. This algorithm resulted in the sensor placement displayed in Figure 5.2(a). Six of the eight sensors were placed in the right-side of the core, while the remaining two were spread out evenly into the FPAdd and I-cache.

The thermal-gradient aware k-means clustering algorithm was implemented on the



(a) 8-Core sparse architecture maximum temperatures folded onto a single chip

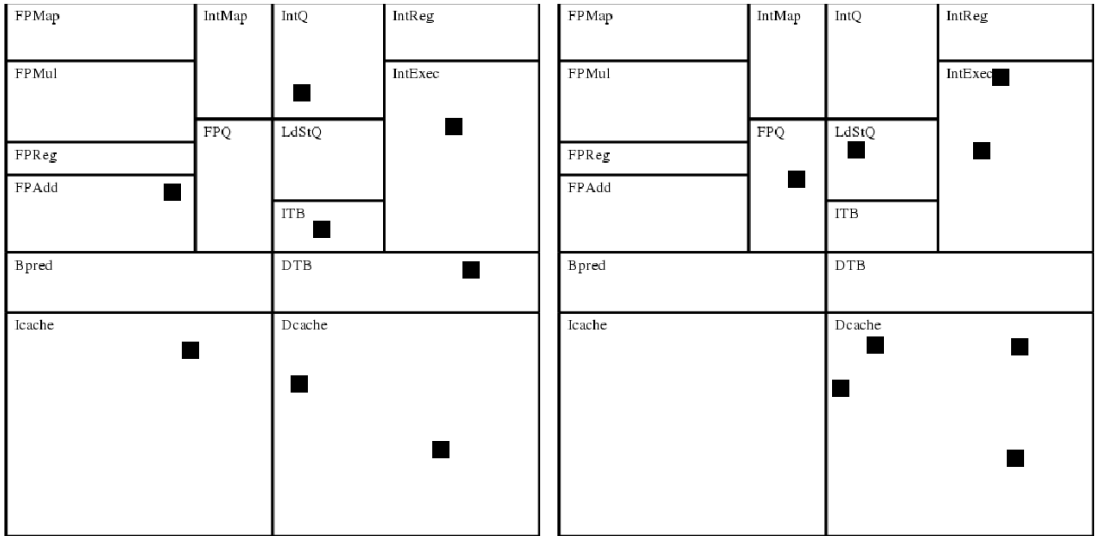
(b) Maximum core temperatures from the 8-core sparse architecture folded onto a single core

Figure 5.1: Thermal maps of the 8-core sparse architecture maximum temperatures.

same folded maximum temperatures core. This algorithm placed four sensors in the D-cache, two in the IntExec, and two in the load-store queue and floating-point queue. The remaining functional blocks were left without any nearby thermal sensors.

The stochastic non-uniform subsampling algorithm was used on the same core thermal map to produce the sampling locations shown in Figure 5.3(a). These samples were then clustered using the basic k-means approach, producing the sensor locations displayed in Figure 5.3(b). Three sensors were placed in the D-cache, the hottest unit. The remaining five sensors were placed in the Icache, FPReg, IntQ, integer register file, and IntExec. Though these five sensors were not necessarily placed near areas of significant temperature, they were placed throughout the core to accurately represent the thermal gradient.

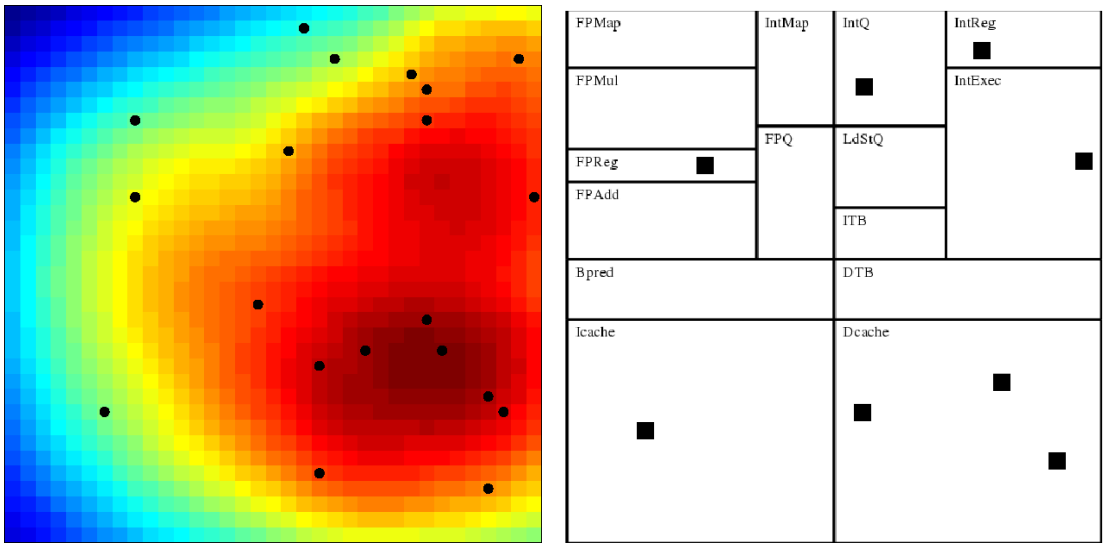
The results from simulating the sensor placement schemes on the sparse architecture are displayed in Table 5.1. Two measurements are reported for each test set on the sparse architecture. Mean error refers to the mean difference between all true temperatures given by HotSpot and the interpolated temperature estimation calculated from the known sensor



(a) Basic k-means clustering sensor placement

(b) Thermal-gradient aware k-means clustering sensor placement with a = 2500

Figure 5.2: Sensor placement using k-means clustering on a single core in the 8-core sparse architecture.



(a) Samples taken using stochastic non-uniform subsampling

(b) Sensor placement via basic k-means clustering of sample points

Figure 5.3: Non-uniform subsampling with k-means clustering sensor placement on a single core in the 8-core sparse architecture.

temperatures for all cores in each architecture. Maximum error refers to the largest error in estimated temperature observed in a core for each sensor configuration. Both linear and cubic interpolation were used in temperature estimation between the sensors. Positive temperature errors indicate that the sensor placement and reconstruction scheme combination resulted in overestimated temperatures, while negative temperature errors indicate an underestimate.

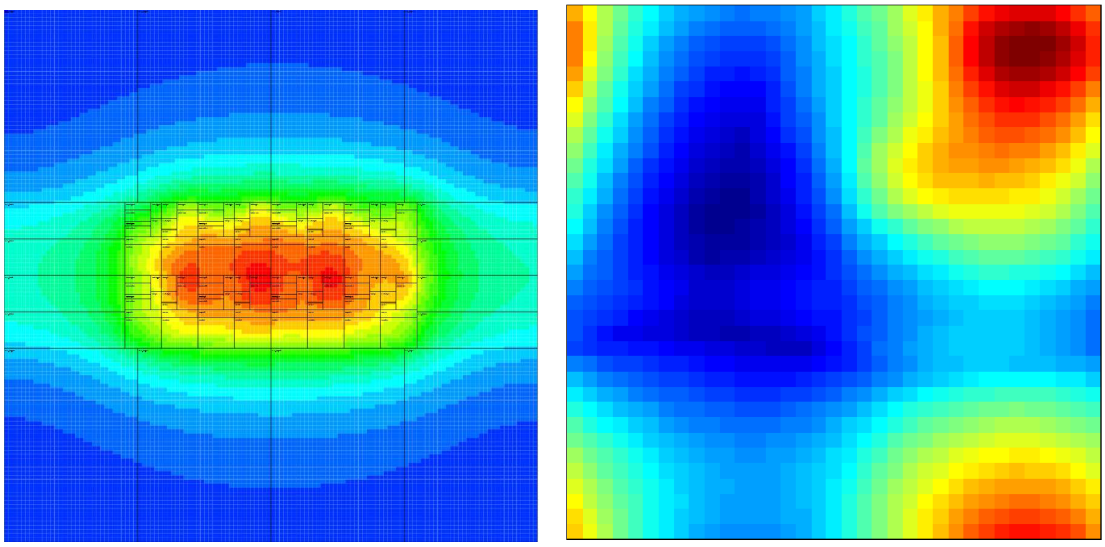
The thermal-gradient aware k-means clustering sensor placement had an improvement in all errors over basic k-means clustering sensor placement for the sparse architecture. Non-uniform subsampling showed the lowest average error and lowest maximum error for both test sets and both interpolation methods. All three sensor placement algorithms resulted in maximum errors of at least 9.09°C overestimate. These overestimates were observed in the cooler regions of each core. No overestimates were high enough to be considered a false thermal emergency (greater than or equal to 82°C).

Sensor Placement Method	**Test Set**	**Interpolation Method**	**Mean Error**	**Maximum Error**	**Improvement**
Basic K-means	Set 1	Linear	1.29 °C	14.86 °C	-
	Set 2	Linear	1.44 °C	13.75 °C	-
	Set 1	Cubic	1.38 °C	15.7 °C	-
	Set 2	Cubic	1.84 °C	13.75 °C	-
Thermal-Gradient Aware K-means	Set 1	Linear	1.02 °C	13.51 °C	20%
	Set 2	Linear	1.16 °C	14.04 °C	19%
	Set 1	Cubic	0.98 °C	14.36 °C	28%
	Set 2	Cubic	1.17 °C	13.59 °C	35%
Non-Uniform Subsampling	Set 1	Linear	-0.31 °C	9.09 °C	76%
	Set 2	Linear	-0.29 °C	10.16 °C	79%
	Set 1	Cubic	-0.17 °C	9.24 °C	88%
	Set 2	Cubic	-0.18 °C	9.55 °C	**90%**

Table 5.1: 8-Core sparse architecture thermal reconstruction results, with minimum errors in boldface text.

Figure 5.4(a) shows the maximum observed temperatures for the 8-core dense architecture folded onto a single chip. Each core was thermally influenced by adjacent cores. The maximum core temperatures from the 8-core dense architecture were folded onto a

single core. A thermal gradient map of the resulting temperatures is shown in Figure 5. The hottest functional block for the dense architecture was the integer register file. The lateral heat propagation from this functional block created significant temperatures in other functional blocks in adjacent cores.



(a) 8-Core dense architecture maximum temperatures folded onto a single chip

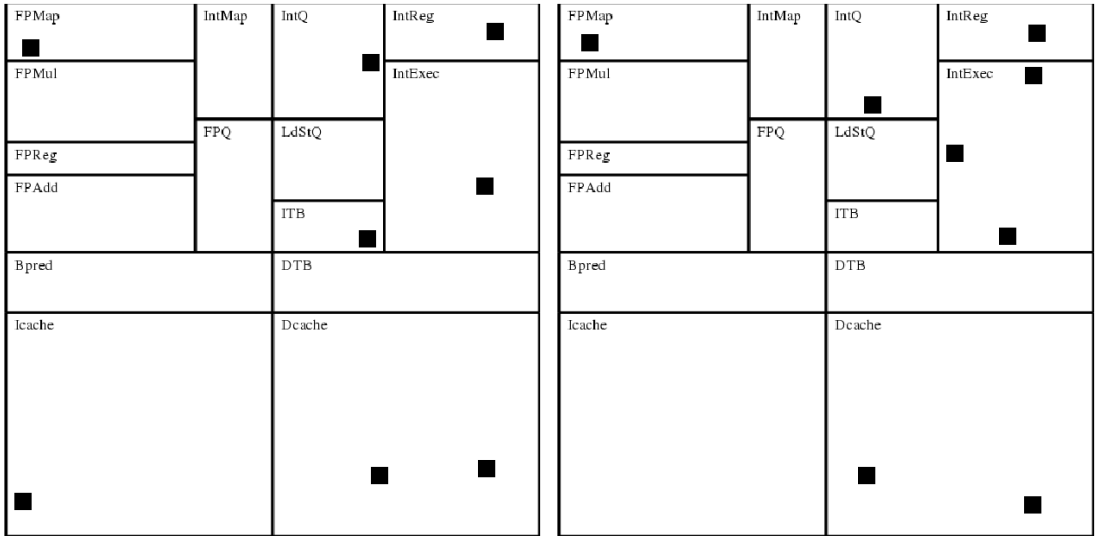
(b) Maximum core temperatures from the 8-core dense architecture folded onto a single core

Figure 5.4: Thermal maps of the 8-core sparse architecture maximum temperatures.

The basic k-means clustering algorithm was implemented on the folded maximum temperatures core from the 8-core dense architecture for eight thermal sensors. This algorithm resulted in the sensor placement displayed in Figure 5.5(a). Four of the eight sensors were placed in the upper right corner near the integer functional blocks. The other sensors covered the remaining corners of the core, leaving the center of the core without any sensors.

The thermal-gradient aware k-means clustering algorithm was also implemented on the same folded maximum temperatures core. this algorithm placed five sensors near the integer functional blocks. Two sensors were placed in the bottom of the D-cache and one sensor was placed in the FPMap. The hot spots in the I-cache were left without nearby sensors.

The stochastic non-uniform subsampling algorithm was used on the same core thermal



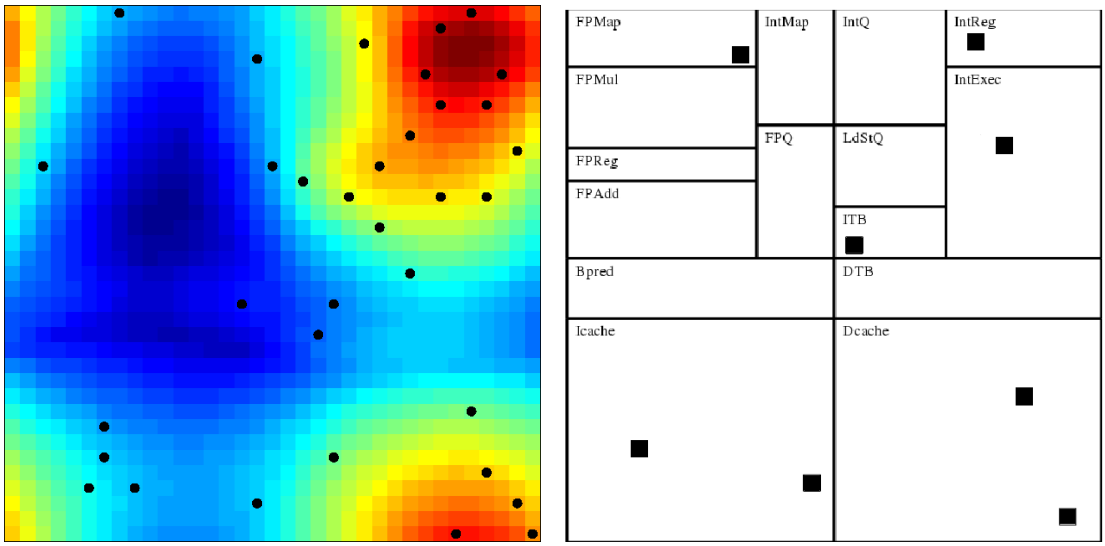
(a) Basic k-means clustering sensor placement

(b) Thermal-gradient aware k-means clustering sensor placement with a = 2500

Figure 5.5: Sensor placement using k-means clustering on a single core in the 8-core dense architecture.

map from the dense architecture to produce the sampling locations displayed in Figure 5.6(a). These samples were then clustered using the basic k-means approach, producing the sensor locations shown in Figure 5.6(b). Three sensors were placed in the integer functional blocks, the hottest units in the core. Two sensors were placed in the D-cache, another considerably hot functional block. Three of the remaining sensors were placed in the other two corners of the core, covering the lateral heat propagation from the integer register file in adjacent cores. A single sensor was placed near the center of the core to measure the expected cooler temperatures.

The results from simulating the sensor placement schemes on the dense architecture are displayed in Table 5.2. Error measurements are defined identically as for Table 5.1. The non-uniform subsampling with k-means clustering sensor placement had an improvement in all errors over both k-means clustering sensor placement mechanisms. The largest errors encountered in non-uniform subsampling were underestimates, while both k-means algorithms had overestimates.



(a) Samples taken using stochastic non-uniform subsampling

(b) Sensor placement via basic k-means clustering of sample points

Figure 5.6: Non-uniform subsampling with k-means clustering sensor placement on a single core in the 8-core dense architecture.

The thermal-gradient aware k-means clustering sensor placement on the dense architecture had a much greater mean error and maximum error for both test sets and both interpolation methods, while both basic k-means clustering and non-uniform subsampling errors noticeably improved for the dense architecture over the sparse architecture.

The thermal-gradient aware k-means clustering results are consistent with those encountered in [41]. This algorithm was more successful in the sparse architecture because each core is thermally insulated with the L2-cache. The thermal patterns within each core were very similar, unlike the core patterns in the dense architecture. In the dense architecture, the cores thermally influence each other much more easily. Using a sensor placement mechanism that places sensors only near hot spots is not as effective for dense architectures because not all cores will have similar hot spots. A more conservative sensor placement mechanism, such as non-uniform subsampling, is better suited for a dense architecture.

Based on the observed temperature estimation errors in both architectures, it can be concluded that estimating temperature is much more accurate in dense architectures when

Sensor Placement Method	Test Set	Interpolation Method	Mean Error	Maximum Error	Improvement
Basic K-means	Set 1	Linear	-0.52 °C	3.8 °C	-
	Set 2	Linear	-0.50 °C	3.81 °C	-
	Set 1	Cubic	-0.34 °C	3.13 °C	-
	Set 2	Cubic	0.38 °C	3.18 °C	-
Thermal-Gradient Aware K-means	Set 1	Linear	-2.58 °C	-9.20 °C	-
	Set 2	Linear	-2.98 °C	-10.74 °C	-
	Set 1	Cubic	-2.46- °C	-11.02 °C	-
	Set 2	Cubic	-2.90 °C	-8.15 °C	-
Non-Uniform Subsampling	Set 1	Linear	-0.07 °C	-3.41 °C	86%
	Set 2	Linear	-0.06 °C	-3.42 °C	88%
	Set 1	Cubic	0.05 °C	-2.94 °C	85%
	Set 2	Cubic	0.04 °C	-2.96 °C	**89%**

Table 5.2: 8-Core dense architecture thermal reconstruction results, with minimum errors in boldface text.

the processing cores are located side-by-side. When the cores are placed sparsely throughout the chip, sensor measurements from one core are not very helpful in determining the thermal profile of a separate core within the same processor. Sensors in cores with adjacent sides in dense architectures are much more useful in determining a single core's thermal profile due to their closer proximity.

Chapter 6

Sensor Mini-Network

Sensor networks on chip for the purpose of thermal monitoring have not been researched extensively in the past. The work in [70] focuses on network interfaces and routing in a Monitor Network on a Chip (MNoC) architecture for multi-core processor. This work does not focus on improving thermal map data recovery, but on data collection, network latency minimization, and area cost reduction.

Use of an MNoC architecture has been proposed for multi-core processor thermal monitoring in [69] with focus on thermal sensor placement. Two different approaches were analyzed:

- **Regular MNoC structure** Modify the basic k-means clustering algorithm to treat the MNoC interfaces as special hotspots with fixed locations for the purpose of reducing wire length and therefore latency.
- **Flexible MNoC structure** Execute sensor placement without concern of the MNoC interfaces. Subsequently cluster the sensors using the basic k-means clustering approach and place the MNoC interfaces at the centroid of each cluster.

6.1 Sensor Mini-Network Configuration

In order for exascale computing systems to become a reality, several hundred thousand cores will be combined into a single system. VLSI circuits of this scale are very prone to

temperature-related reliability concerns, thus maximum thermal coverage with a minimal number of sensors is of utmost importance. Using a large number of sensors for coverage of a single core is too complicated for kilocore systems and will incur excessive overheads in terms of area and power. For this reason, a minimum number of thermal sensors will be placed on each core at points that are most beneficial for capturing the core's thermal activity at run-time.

The sensors will be arranged in a Sensor Mini-Network (SMN) for the purpose of sensing the complete thermal profile of a chip. A 64-homogeneous core version of the SMN is depicted in Figure 6.1. Each sensor periodically reports its measured temperature to a Reliability Unit so that it can be used in conjunction with temperatures from other sensors to ultimately determine thermal information for locations on the chip that are not directly measured by the sensors. The SMN configuration is more accurately able to sense thermal violations across the entire chip than the same number of sensors using no communication.

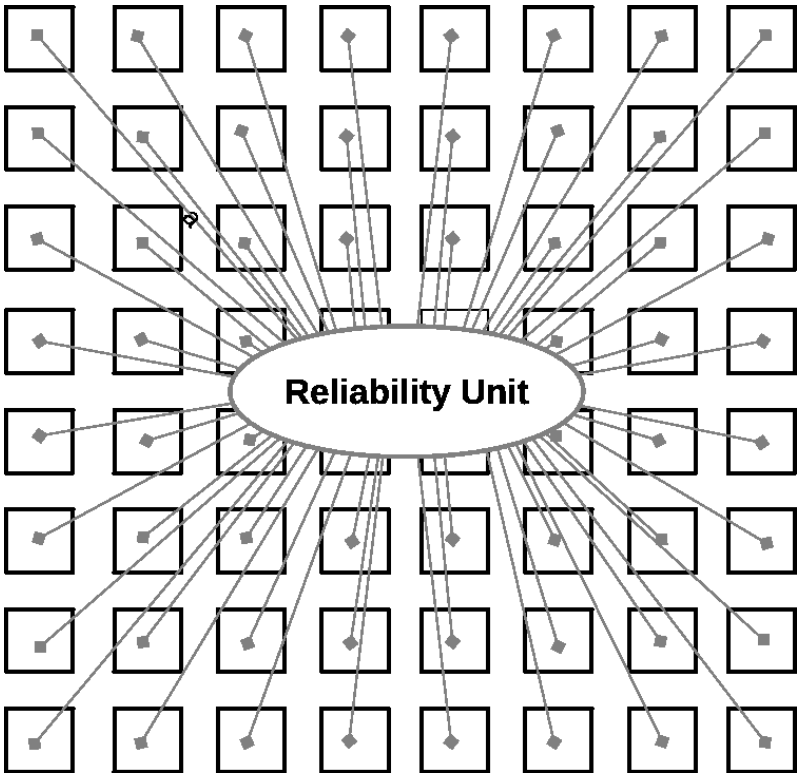


Figure 6.1: Sensor Mini-Network configuration with 64 homogeneous cores.

To determine the appropriate number of bits required to encode all temperature data sufficiently, the minimum and maximum expected temperatures for a specific architecture must be considered according to Equation 6.1. The difference between the maximum temperature T_{max} and the minimum temperature T_{min} (set to the ambient temperature) is divided by the desired resolution in Kelvin. This calculation give the necessary number of codes to be used, from which the required number of bits, n, can be calculated.

$$Number\ of\ codes = \frac{T_{max} - T_{min}}{Resolution} \tag{6.1}$$

For the following discussions, consider a 1024-core architecture modeled at 25 nm technology. A representative trace of typical thermal operation was obtained for each of the 1024 cores running a randomly selected benchmark from Table 4.4. Thermal simulations were conducted using HotSpot 5.0 at a grid size of 1024 x 1024. The corresponding thermal map is displayed in Figure 6.2. A uniform grid of sensors placed evenly across the die is assumed for simplicity. Histograms of the temperatures seen at these sensor locations are shown in Figures 6.3(a) and 6.3(b) for a total of 2048 thermal sensors (Configuration A) and a second scenario of 1024 sensors (Configuration B). These histograms indicate all possible expected temperature measurements that will need to be represented in the SMN. For both configurations in the 1024-core architecture, all observed temperatures were under 400 *K*.

Basic linear interpolation was used to determine temperatures for locations without thermal sensors for the two different sensor configurations. The distributions of error in estimation for all points in the architecture for the two configurations are displayed in Figures 6.4(a) and 6.4(b). Configuration A uses 2048 thermal sensors and has 87% of error less than 1 *K* and 100% of error less than 3 *K*. Configuration B uses 1024 total sensors and has 73% of error less than 4 *K* and 100% less than 6 *K*.

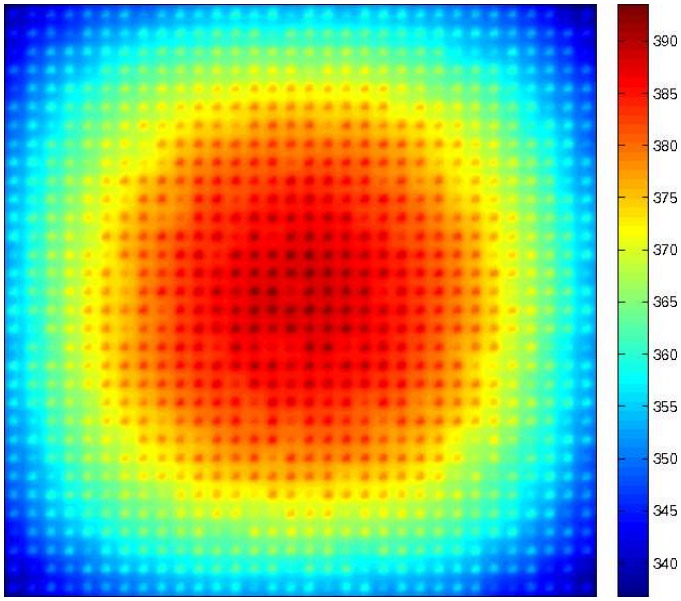
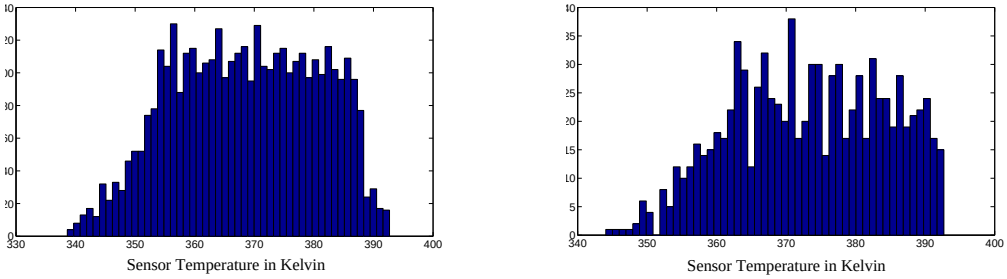


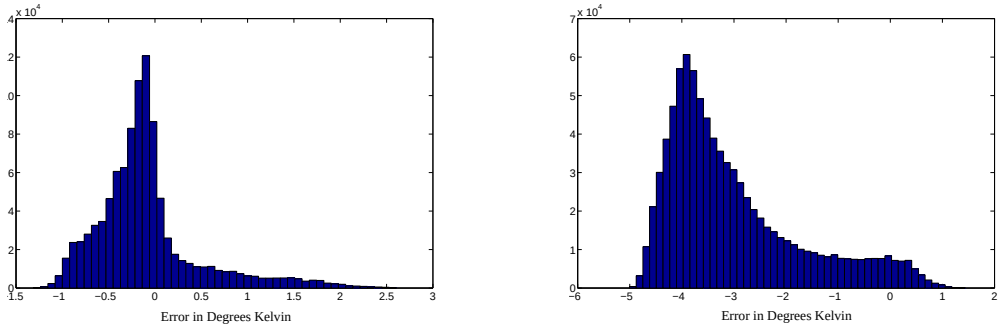
Figure 6.2: Steady-state thermal map of a homogeneous 1024-core architecture.



(a) Accumulated sensor readings from using 2048 sensors (Configuration A).

(b) Accumulated sensor readings from using 1024 sensors (Configuration B).

Figure 6.3: Accumulated temperature sensor readings in the 1024-core sparse architecture.



(a) Error in estimation from using 2048 Sensors (Configuration A).

(b) Error in estimation from using 1024 Sensors (Configuration B).

Figure 6.4: Magnitudes of all temperature estimation errors for the 1024-core architecture.

6.1.1 Baseline SMN [No Compression]

The most straight-forward SMN configuration is to have each thermal sensor report its temperature individually to the Reliability Unit without compressing any data. There is no communication between sensors; all sensors are handled identically. Linear interpolation between the sensors is used to determine the thermal data for the entire chip, similar to the interpolation-based uniform strategy discussed previously.

To determine the number of bits required to represent all possible temperatures in an architecture, the maximum and minimum temperatures must be taken into account. For the 1024-core architecture, the ambient temperature for the experiment was set to 318 *K*. This temperature is the minimum for the architecture. All observed temperatures were under 400 *K*. At a resolution of 1 *K*, this range requires a minimum of 82 codes. This number of codes can be represented in a minimum of n = seven bits. Seven bits allow 128 quantization levels Q_0 through Q_{127} that each represent a temperature in the range of 1 *K*. Each code is assigned the value in the middle of the range in order to minimize the mean squared error. This representation is displayed in Table 6.1.

Temperature Range	7-bit Code	Q-level	Quantized Temperature
317.5 K to 318.5 K	000 0000	Q_0	318
318.5 K to 319.5 K	000 0001	Q_1	319
319.5 K to 320.5 K	000 0010	Q_2	320
.	.	.	.
.	.	.	.
.	.	.	.
443.5 K to 444.5 K	111 1111	Q_{127}	444

Table 6.1: Temperature quantization levels for the 1024-core architecture.

6.1.2 SMN Differential Encoding

To reduce communication bandwidth and overall power consumption, the amount of transmitted data should be minimized. In the differential encoding algorithm, some sensors are assigned to be reference sensors and the remaining are node sensors. Assuming that thermal sensors have been uniformly placed in a grid throughout the die, every-other sensor will be a reference sensor. A sample setup of nine sensors is shown in Figure 6.5. Sensors $S0, S2, S6$, and $S8$ are references sensors while $S1, S3, S4, S5$, and $S7$ are node sensors.

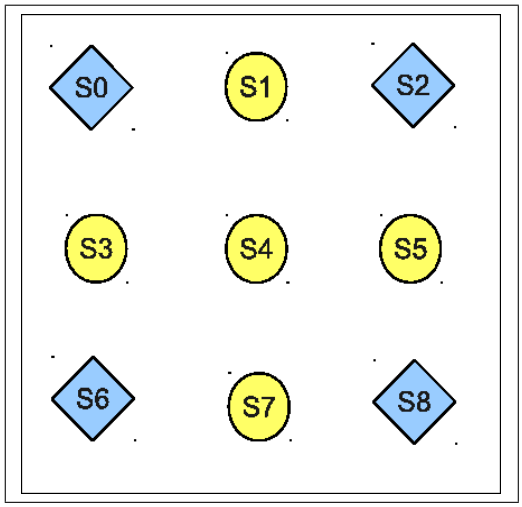


Figure 6.5: Uniform grid of reference and node sensors.

The SMN differential encoding setup is displayed in Figure 6.6. The reference sensors sample and digitize their uncompressed temperature $T_{reference}$ and send the result to the Reliability Unit. In the 1024-core architecture example, the reference sensors send the

uncompressed 7-bit temperature representation as displayed in Table 6.1. The reference sensors located closest to each node sensor can be used to reduce the number of bits needed to represent the temperature at each node sensor. Each node sensor is assumed to be able to receive the two closest located reference sensors' uncompressed transmissions. From these two transmissions, an estimate temperature for the node sensor is computed using basic linear interpolation. The difference between this estimate and the node sensor's true temperature T_{node} is computed, compressed, and transmitted to the Reliability Unit.

The node sensor temperature represented as a compressed codeword will be sent to the Reliability Unit where it will be decoded into a temperature difference. This decoded difference will subsequently be added to the temperature estimate from the corresponding reference sensors in order to recover the true temperature for this node sensor.

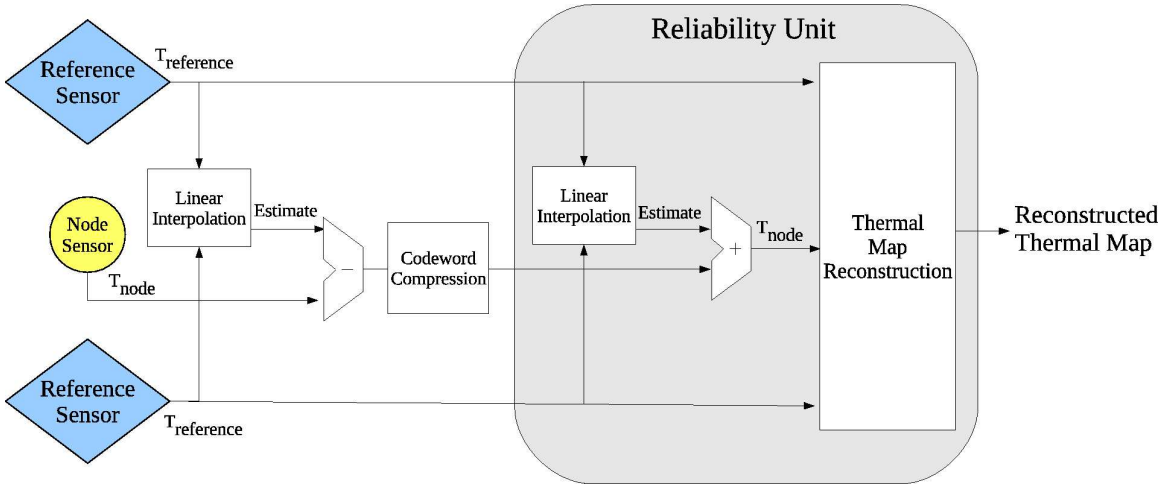
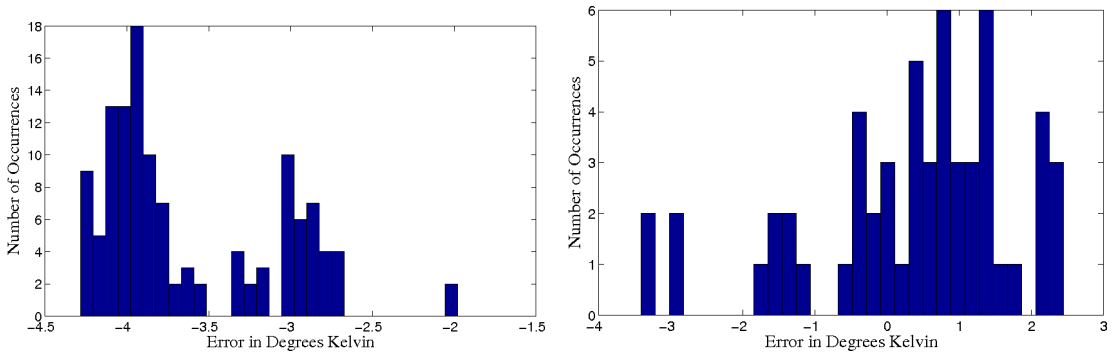


Figure 6.6: SMN differential encoding block diagram.

To determine the fewest possible number of bits required for the node sensor codewords at the specified resolution, statistical data for the given architecture must be gathered surrounding the accuracies of linearly estimating solely using reference sensors.

For a given typical trace of thermal data, the reference sensors can be used to linearly estimate the temperature values at each of the node sensor locations. If a uniform grid of sensors configuration is assumed as in Figure 6.5, data for all node sensors can be combined and analyzed together. In this configuration, all node sensors are located at the same

distance from their corresponding reference sensors. Histograms of the errors for the 1024-core architecture using 1024 reference sensors and 512 reference sensors are shown in Figures 6.7(a) and 6.7(b).



(a) Temperature estimation error for Configuration A, 1024 Reference Sensors.

(b) Temperature estimation error for Configuration B, 512 Reference Sensors.

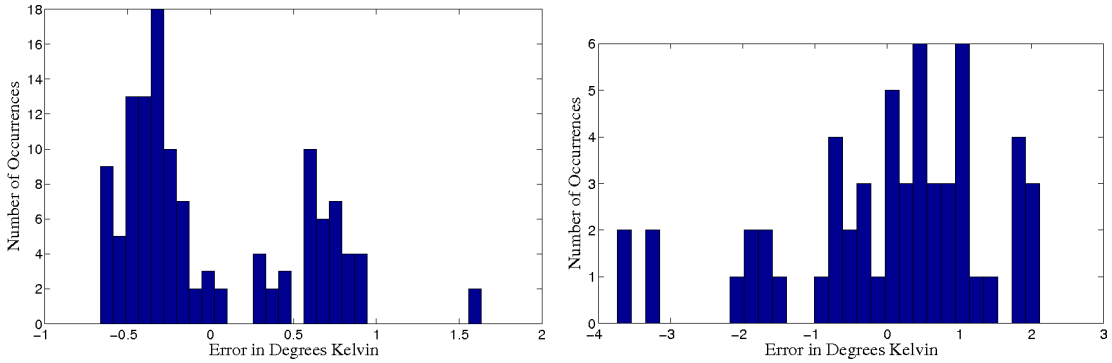
Figure 6.7: Temperature estimation error histograms at node sensor locations in the 1024-core architecture

The mean of the gathered errors should be equal to zero. To ensure that this is the case, the temperature estimation T can be offset by the true mean according to Equation 6.2, where $\bar{e}(x, y)$ is equal to the true mean of the error at a distance (x, y) from the reference sensors. Using this equation with the original histograms from Figures 6.7(a) and 6.7(b) generates the updated histograms in Figures 6.8(a) and 6.8(b).

$$\hat{T}(x, y) = \alpha(x, y)T(x_1, y_1) + \beta(x, y)T(x_2, y_2) + \bar{e}(x, y) \tag{6.2}$$

Gathering the errors in estimation for all node sensor locations yields a fixed range of possible estimation errors. This range divided by the desired resolution gives the number of codewords required and the necessary number of bits via Equation 6.1. All node sensor temperatures will be encoded into these compressed codewords. Several different quantization temperature levels are assigned to the same codeword.

For the 1024-core architecture with 1024 reference sensors and 1024 node sensors, it can be deduced from Figure 6.8(a) that a minimum of four codewords are needed at a resolution of 1 *K*, which can be represented in two bits. For the scenario with 512 reference



(a) Temperature estimation error with adjusted mean for Configuration A, 1024 Reference Sensors.

(b) Temperature estimation error with adjusted mean for Configuration B, 512 Reference Sensors.

Figure 6.8: Temperature estimation error histograms with adjusted mean at node sensor locations in the 1024-core architecture

sensors and 512 node sensors, Figure 6.8(b) shows that a minimum of seven codewords are needed at the same resolution, which can be represented in a minimum of three bits. Tables 6.2 and 6.3 display a more detailed summary of codeword assignments for these two scenarios.

Temperature Difference Range	**2-bit Codeword**
-1.5 *K* to -0.5 *K*	00 = C_0
-0.5 *K* to 0.5 *K*	01 = C_1
0.5 *K* to 1.5 *K*	10 = C_2
1.5 *K* to 2.5 *K*	11 = C_3

Table 6.2: 2-Bit codewords for SMN differential encoding compression in the 1024-core architecture.

Table 6.4 displays the performance results for the 1024-core architecture using both sensor configuration scenarios. Using differential encoding to compress node sensor temperatures in both sensor configurations significantly improves the number of transmitted bits to the Reliability Unit over a no compression scenario without losing any information.

Both Configuration A and Configuration B significantly improved the number of transmitted bits to the Reliability Unit. Configuration A, only using 2-bit codewords, improved upon the no compression scheme by 36%. Conifguration B, using 3-bit codewords, improved upon the no compression scheme by 29%. Both configurations could have used

Temperature Difference Range	3-bit Codeword
-4.5 *K* to -3.5 *K*	000 = C_0
-3.5 *K* to -2.5 *K*	001 = C_1
-2.5 *K* to -1.5 *K*	010 = C_2
-1.5 *K* to -0.5 *K*	011 = C_3
-0.5 *K* to 0.5 *K*	100 = C_4
0.5 *K* to 1.5 *K*	101 = C_5
1.5 *K* to 2.5 *K*	110 = C_6
2.5 *K* to 3.5 *K*	111 = C_7

Table 6.3: 3-Bit codewords for SMN differential encoding compression in the 1024-core architecture.

Scheme	Reference Sensors	Node Sensors	Bits Transmitted to Reliability Unit	Max Error	Improvement
No compression	2048	0	14,336	3 *K*	-
2-bit Diff. Encoding	1024	1024	11,980	3 *K*	36%
No compression	1024	0	7,168	6 *K*	-
3-bit Diff. Encoding	512	512	5,120	6 *K*	29%

Table 6.4: SMN differential encoding performance results.

fewer bits for compressed codewords to achieve a further improvement in performance at the cost of more erroneous decoding. Both configurations in this example were assigned codeword sizes to account for the full histogram of error possibilities, however, an unforeseen outlying temperature difference could occur and cause incorrect temperature recovery. This trade-off must be considered when designing the SMN with differential encoding.

By only sending the temperature differences from the reference sensor, the number of bits required by the node sensors could be greatly reduced. The node sensors will be located within the range of a maximum known distance from the reference sensors. This specification directly defines the maximum possible temperature differential that can physically occur between a node sensor and a corresponding reference sensor. This maximum possible temperature differential corresponds to the maximum number of bits that will be needed to accurately represent all possible observed differences. The number of bits required is significantly smaller than that required to represent the full temperature value, thus saving on power consumption and bandwidth to the Reliability Unit.

Though the number of transmitted bits to the Reliability Unit is significantly reduced through differential encoding, additional power is required for the reference sensor temperatures to reach the node sensors. To eliminate this cost and reduce routing complexity, compression through distributed source coding was explored.

6.1.3 SMN Distributed Source Coding

A second method of compressing the temperature data and reducing the number of bits to transmit can be achieved by using counters and eliminating communication between the reference and node sensors. The node sensor temperatures can still be compressed without losing any vital thermal information at their locations. The reference sensors send uncompressed data to the Reliability Unit while the node sensors are compressed using distributed source coding (DSC). Figure 6.9 shows a block diagram of SMN with DSC for two reference senors and one node sensor. Each sensor sends its compressed or uncompressed temperature directly to the Reliability Unit where the compressed node temperatures T_{node} are decoded into original temperatures.

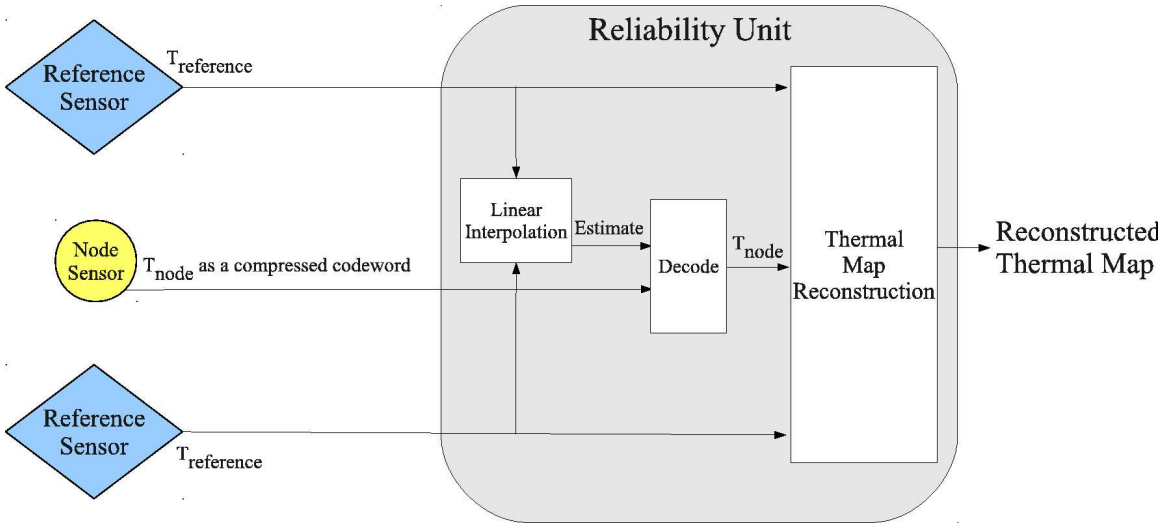


Figure 6.9: SMN distributed source coding block diagram.

As displayed in Figure 6.10, the temperature measured by a sensor is compared with the output of a counter. When the comparator indicates that the two values are equal,

the counter will be stopped and its current output value will be recorded as the encoded temperature for that sensor. To reduce the number of bits required, the reference sensor will use an n-bit counter and n bits to represent the transmitted reference temperature, $T_{reference}$. Each node sensor uses m fewer bits to represent the temperature. The node sensors use a smaller $n - m$-bit counter and an $n - m$-bit codeword to represent their transmitted temperature.

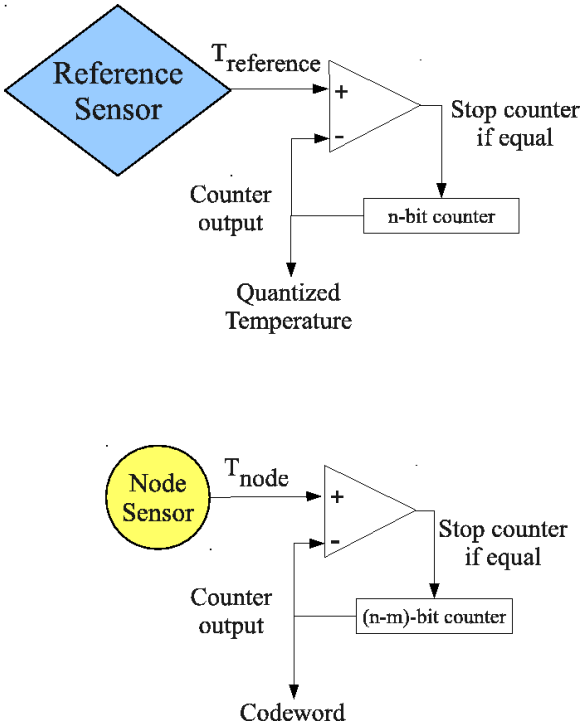


Figure 6.10: SMN distributed source coding sensor counters.

To determine the fewest possible number of bits required for node sensor codeword at the specified resolution, statistical data for the given architecture must be gathered surrounding the accuracies of linearly estimating solely using reference sensors, as done in SMN differential encoding. The same process of determining the maximum error in temperature estimation at all node sensor locations can be used for SMN distributed source coding. The error histograms are the same in both compression schemes because use the same linear estimation from the same thermal trace data.

For the 1024-core architecture with 1024 reference sensors and 1024 node sensors

(Configuration A), it can be deduced from the SMN differential encoding codeword size determination process that a minimum of four codewords are needed at a resolution of 1 *K*, which can be represented in two bits. For the scenario with 512 reference sensors and 512 node sensors (Configuration B), Figure 6.8(b) shows that a minimum of seven codewords are needed at the same resolution, which can be represented in a minimum of three bits. In SMN, these codewords represent the actual temperature measured by the node sensors, rather than the difference between the estimate and the true temperature. Table 6.5 displays a more detailed summary of codeword assignments and representations for these two scenarios. The Q-levels correspond to the same 7-bit representations displayed previously in Figure 6.1.

Temperature Range	**2-bit Codeword**	**3-bit Codeword**	**Q-level**	**Quantized Temperature**
317.5 *K* to 318.5 *K*	00 = C_0	000 = C_0	Q_0	318
318.5 *K* to 319.5 *K*	01 = C_1	001 = C_1	Q_1	319
319.5 *K* to 320.5 *K*	10 = C_2	010 = C_2	Q_2	320
320.5 *K* to 321.5 *K*	11 = C_3	011 = C_3	Q_3	321
321.5 *K* to 322.5 *K*	00 = C_0	100 = C_4	Q_4	322
322.5 *K* to 323.5 *K*	01 = C_1	101 = C_5	Q_5	323
323.5 *K* to 324.5 *K*	10 = C_2	110 = C_6	Q_6	324
324.5 *K* to 325.5 *K*	11 = C_3	111 = C_7	Q_7	325
325.5 *K* to 326.5 *K*	00 = C_0	000 = C_0	Q_8	326
326.5 *K* to 327.5 *K*	01 = C_1	001 = C_1	Q_9	327
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
443.5 *K* to 444.5 *K*	11 = C_3	111 = C_7	Q_{127}	444

Table 6.5: Codewords for SMN distributed source compression in the 1024-core architecture.

From each node sensor, the Reliability Unit receives a compressed codeword. To decipher which of the temperatures associated with a codeword is the true node sensor temperature, each Decode block in the Reliability Unit must compare the originally estimated temperature for the node sensor's location with the codeword.

Figure 6.11 illustrates an example of this comparison process for the 1024-core architecture with 1024 reference sensors at 1024 node sensors represented with 2-bit error codewords (Configuration A). A node sensor measures 342.6 *K*, which falls in the range 342.5 *K* to 343.5 *K* and is represented with codeword C_1. The node sensor sends C_1 to the Reliability Unit where it is compared with the estimate for this location. In this case, the estimate is 341 *K*, which is represented with the quantization level Q_{23}. A histogram of the possible error in the estimate is super-imposed over this quantization level, centering the zero-point over Q_{23}. The closest quantization levels to Q_{23} that correspond to a codeword of C_1 are Q_{21} and Q_{25}. Because the error histogram centered at estimate Q_{23} overlaps the ambiguous code Q_{25} and not Q_{21}, the recovered node sensor temperature is quantization level Q_{25}, or 343 *K*.

Table 6.6 displays the performance results for the 1024-core architecture using both sensor configuration scenarios. Using DSC to compress node sensor temperatures in both sensor configurations significantly improves the number of transmitted bits to the Reliability Unit over a no compression scenario without losing any information.

Scheme	**Reference Sensors**	**Node Sensors**	**Bits Transmitted to Reliability Unit**	**Max Error**	**Improvement**
No compression	2048	0	14,336	3 *K*	-
2-bit DSC	1024	1024	11,980	3 *K*	36%
No compression	1024	0	7,168	6 *K*	-
3-bit DSC	512	512	5,120	6 *K*	29%

Table 6.6: SMN distributed source coding performance results.

The improvements in both scenarios are identical to those encountered in SMN differential encoding because the node sensor codeword sizes are based upon the same statistical analysis. This algorithm, however, saves more power and bandwidth than the previously mentioned differential encoding technique because each node sensor does not need to receive the estimated temperature and compute the difference. There is no communication between sensors. The node sensors simply send the output of their counter as it is, regardless of the reference sensors' temperature readings. There is virtually no added complexity

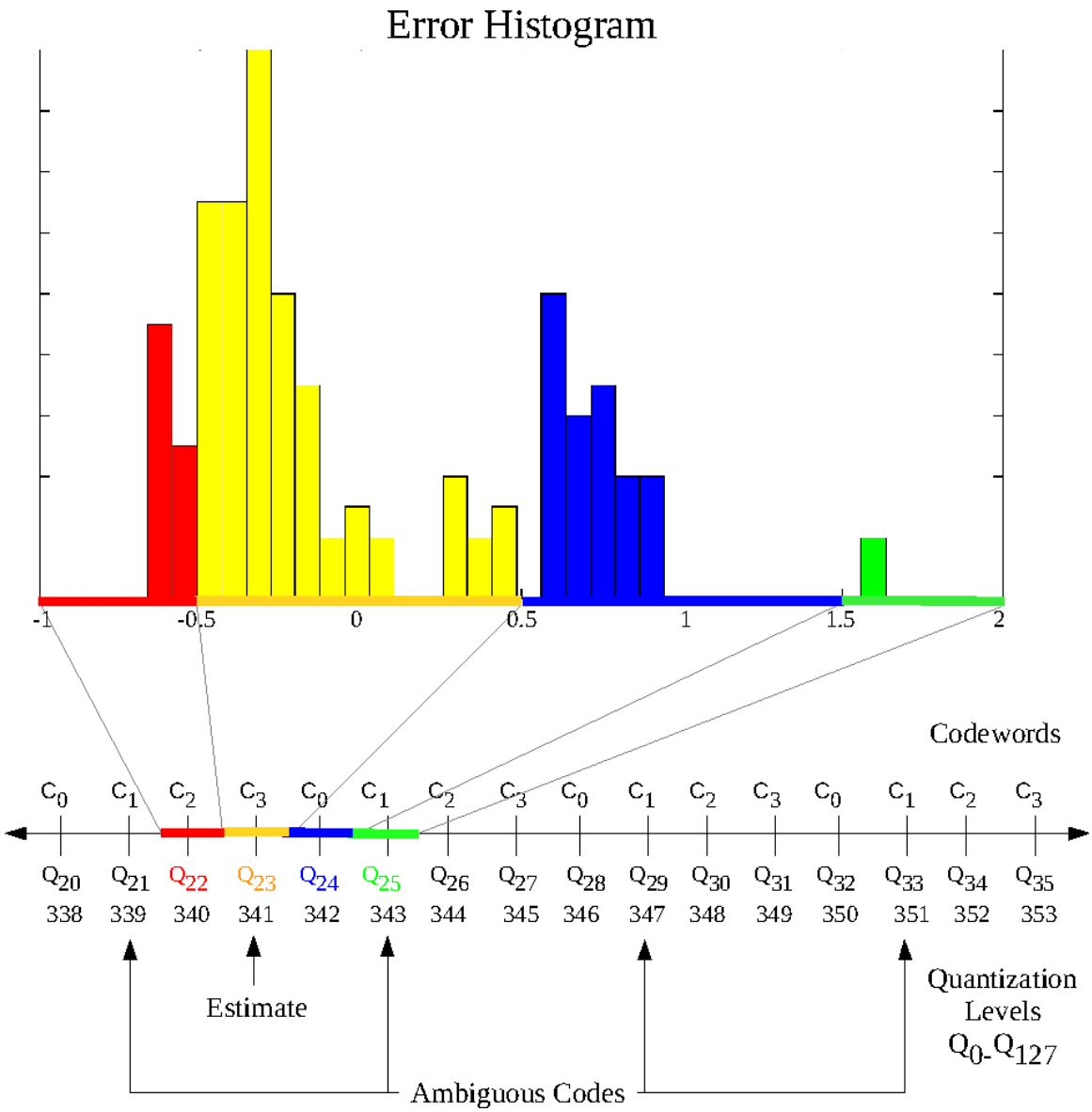
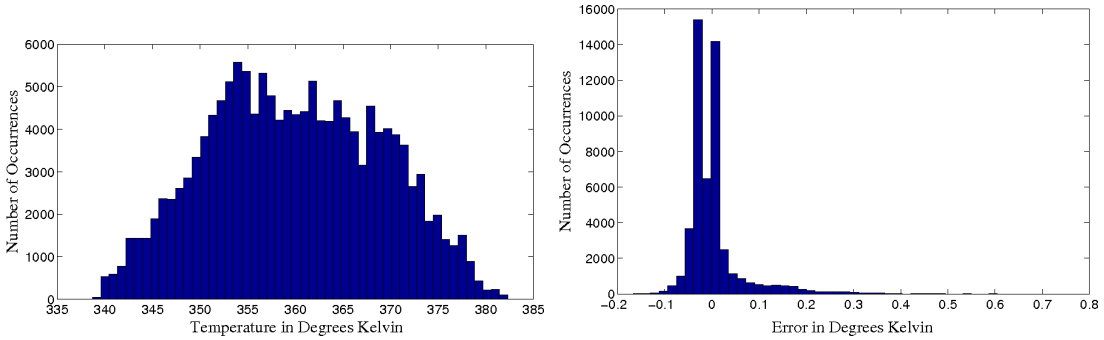


Figure 6.11: SMN distributed source coding example.

(as compared to the no compression scheme) to encode the node sensor codewords. Comparing overheads of the Reliability Unit is beyond the scope of this thesis.

6.2 SMN on a 1024-Core Architecture

The previously discussed compression schemes within an SMN could also be applied to architectures containing many sensors for each core. Consider the 1024-core architecture from Section 6.1 with nine sensors for each core, arranged in a uniform grid. Increasing the number of sensors reflects a more realistic scenario where each core has multiple sensors. The temperatures measured by these sensors are accumulated in Figure 6.12(a). The magnitudes of all errors in temperature estimation through linear interpolation are accumulated in the histogram in Figure 6.12(b). Measuring temperature in the 1024-core architecture was much more accurate with this number of sensors; 95% of the errors were under 0.4 K and 100% of the observed errors were under 1 K.



(a) All measured temperatures in the 1024-core architecture using 9 sensors per core.

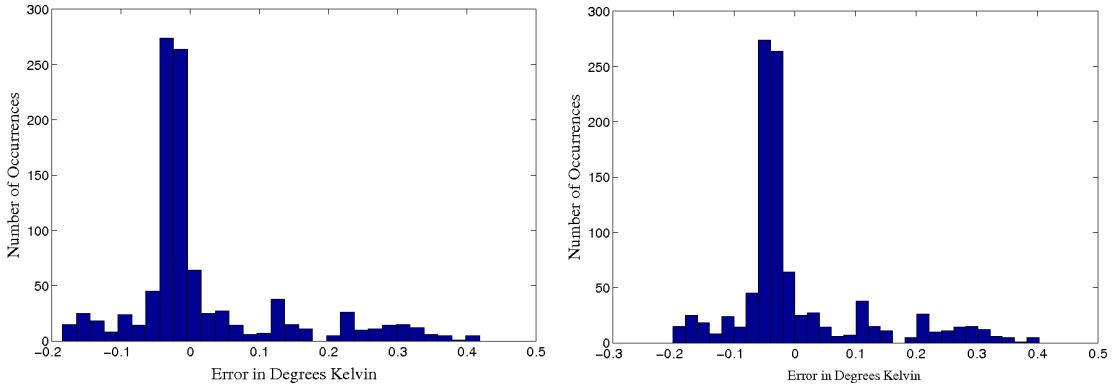
(b) Error in estimation from using 9 sensor for every core.

Figure 6.12: Magnitudes of all temperature estimation errors for the 1024-core architecture.

As seen in the previous sensor configurations of 1024 sensors and 2048 sensors, all observed measured temperatures were under 400 K. The ambient temperature for the experiment was still set to 318 K, so this sensor configuration will use the same 7-bit code representation previously displayed in Table 6.1.

For differential encoding and DSC, the same uniform grid of reference sensors and

node sensors will be used for each core. To compress the node sensor temperatures, the temperature estimation error histogram for each node sensor location in Figure 6.13(a) and its adjusted mean histogram in Figure 6.13(b) reveal a maximum expected error of 0.5 *K*. If a resolution of 1 *K* is specified, then a maximum of one bit is needed for each node sensor. Tables 6.7 and 6.8 display the temperature error range assignments for this configuration.



(a) Temperature estimation error for the node sensors.

(b) Temperature estimation error with adjusted mean for the node sensors.

Figure 6.13: Temperature estimation error histograms at node sensor locations in the 1024-core architecture

Temperature Difference Range	**1-bit Codeword**
-0.5 *K* to 0.5 *K*	0 = C_0
0.5 *K* to 1.0 *K*	1 = C_1

Table 6.7: 1-Bit codewords for SMN differential encoding node sensor compression in the 1024-core architecture.

Table 6.9 displays the performance results from using nine sensors per core. Using compressed codewords significantly improves the number of transmitted bits to the Reliability Unit over a no compression scenario without losing any information.

Again, both differential encoding and DSC obtain identical improvement over a no-compression scheme when observing the number of transmitted bits to the Reliability Unit. The number of transmitted bits in this configuration of nine sensors per core is much greater than that of Configurations A and B discussed in Section 6.1 due to the significant increase in the number of sensors in the network. This magnitude of sensors, however, shows a

DSC Temperature Range	1-bit Codeword	Q-level	Quantized Temperature
317.5 *K* to 318.5 *K*	0 = C_0	Q_0	318
318.5 *K* to 319.5 *K*	1 = C_1	Q_1	319
319.5 *K* to 320.5 *K*	0 = C_0	Q_2	320
.	.	.	.
.	.	.	.
.	.	.	.
443.5 *K* to 444.5 *K*	1 = C_1	Q_{127}	444

Table 6.8: 1-Bit codewords for SMN DSC node sensor compression in the 1024-core architecture.

Scheme	Reference Sensors	Node Sensors	Bits Transmitted to Reliability Unit	Improvement
No compression	9216	0	64,512	-
1-bit Diff. Encoding	4096	5120	33,792	48%
1-bit DSC	4096	5120	33,792	48%

Table 6.9: SMN differential encoding performance results.

temperature estimation error of less than 1 *K* while the other configurations experienced errors up to 3 *K* and 6 *K*. The trade-off between the overheads incurred by many sensors and allowable temperature estimation error must be considered.

Placing the sensors in a uniform grid resulted in a relatively low margin of error for the 1024-core architecture. The low error in temperature estimation is a result of simulation parameters chosen for this architecture. Due to the physical size of the 1024-core architecture and its required simulation time, the grid size for the HotSpot tool did not have the same resolution per core as the smaller 8-core architectures analyzed in Section 4.1.

6.3 SMN with Reduced Resolution

Further compression can be achieved through representing temperatures at a lower resolution. The previous examples assumed a resolution of 1 *K*. Evaluation of the same examples with a resolution of 2 *K* shows that uncompressed temperatures need to be represented in

42 codewords, or a minimum of 6 bits. Using the same analysis for node sensor compression, Configuration A, using 2048 sensors, is capable of compressing temperatures into 1-bit codewords and achieved a 41% improvement in performance over no compression at this resolution. Configuration B, using 1024 sensors, was able to compress node sensor temperatures in 2 bits. Performance results are summarized in Table 6.10.

Scheme	**Reference Sensors**	**Node Sensors**	**Bits Transmitted to Reliability Unit**	**Improvement**
No compression	2048	0	12,288	-
1-bit Compression	1024	1024	7,168	41%
No compression	1024	0	6,144	-
2-bit Compression	512	512	4,096	33%

Table 6.10: Performance results from 2 *K* resolution in the 1024-core architecture.

Chapter 7

Conclusions

The work in this thesis introduced a systematic thermal sensor allocation scheme for accurate thermal monitoring in multi-core processors. The non-uniform subsampling with k-means clustering mechanism focuses on balancing uniform temperature measurement and thermal emergency detection. This mechanism provided an improved average temperature estimation over other non-uniform sensor placement methods for two different 8-core architectures. Using cubic interpolation, the sensors placed via this mechanism were able to help determine the full thermal map of the chip with an improved average error of 90% over sensor placed with basic-kmeans clustering.

The work in this thesis also explored the use of an on-chip sensor min-network for monitoring temperatures at run-time from many sensors. Temperature data from sensors is sent to a central Reliability Unit for thermal map reconstruction through interpolation. To reduce bandwidth and power requirements, the work in this thesis addressed two data compression schemes to use with an sensor mini-network.

Compression through differential encoding reduced the number of transmitted bits to the Reliability Unit, thus saving on across-chip network traffic. In order to compress using differentials, however, sensor-to-sensor communication was introduced. The additional communication requires additional power and complexity. To eliminate sensor-to-sensor communication, compression through distributed source coding was analyzed.

SMN compression through distributed source coding showed to be the best compression scheme due to no communication between sensors. This scheme was able to reduce the

number of transmitted bits by 36% in the presented example of a 1024-core architecture. Though this scheme adds a level of complexity in the Reliability Unit, overheads are not expected to be costly.

There are several research opportunities available for expansion on this topic. Applying the SMN to a processor with non-uniform sensor placement would allow an improvement in error estimation, but also incur additional overheads and complexity in compressed codeword size determination. This trade-off should be analyzed in further detail.

Additional research could also be conducted in applying the SMN to heterogeneous cores rather than homogeneous cores. Cores with dissimilar sensor placements and thermal patterns would benefit greatly from the advantages of an SMN.

Further work is also required to determine the details of various communication protocols that could be applied to an SMN in this domain. Heterogeneous sensors should be further considered for use in the SMN to measure parameters other than temperature.

Bibliography

[1] A.H. Ajami, K. Banerjee, M. Pedram, and L.P.P.P. van Ginneken. Analysis of non-uniform temperature-dependent interconnect performance in high performance ICs. In *Proceedings of the 38th annual Design Automation Conference*, pages 567–572. ACM, 2001.

[2] J. Altet, a. Rubio, a. Salhi, J.L. Galvez, S. Dilhaire, a. Syal, and a. Ivanov. Sensing temperature in CMOS circuits for thermal testing. *22nd IEEE VLSI Test Symposium, 2004. Proceedings.*, pages 179–184, 2004.

[3] Baltasar Beferull-Lozano, Robert L. Konsbruck, and Martin Vetterli. Rate-distortion problem for physics based distributed sensing. In *Proceedings of the Third International Symposium on Information Processing in Sensor Networks*, page 330, New York, New York, USA, 2004. ACM Press.

[4] S. Borkar. Design challenges of technology scaling. *IEEE Micro*, 19(4):23–29, 1999.

[5] S. Borkar. Platform 2015 : Intel Processor and Platform Evolution for the Next Decade. *Intel*, 2005.

[6] Shekhar Borkar, Tanay Karnik, Siva Narendra, and Jim Tschanz. Parameter variations and impact on circuits and microarchitecture. *Proceedings of the Design Automation Conference*, 64:338–342, 2003.

[7] P. Bratek and A. Kos. Temperature sensors placement strategy for fault diagnosis in-integrated circuits. In *Semiconductor Thermal Measurement and Management, 2001. Seventeenth Annual IEEE Symposium*, page 245–251, 2001.

[8] D. Brooks, R.P. Dick, R. Joseph, and L. Shang. Power, thermal, and reliability modeling in nanometer-scale microprocessors. *IEEE Micro*, pages 49–62, 2007.

[9] D. Brooks and M. Martonosi. Dynamic Thermal Management for High Performance Microprocessors. *Proceedings of the 7th International Symposium on High Performance Computer Architecture*, 2001.

[10] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. *ACM SIGARCH Computer Architecture News*, 28(2):94, 2000.

[11] D. Burger and T.M. Austin. The SimpleScalar Tool Set, Version 2.0. *ACM SIGARCH Computer Architecture News*, 25(3):13–25, 1997.

[12] D. Burger and T.M Austin. SimpleScalar Tutorial. 1997.

[13] K. Chakrabarty, S.S. Iyengar, H. Qi, and E. Cho. Grid coverage for surveillance and target location in distributed sensor networks. *IEEE Transactions on Computers*, page 1448–1453, 2002.

[14] Ryan Cochran and Sherief Reda. Spectral techniques for high-resolution thermal characterization with limited sensor data. *Proceedings of the 46th Annual Design Automation Conference*, page 478–483, 2009.

[15] Tilera Corporation. TILE-Gx Processor Family Product Brief, 2009.

[16] Basab Datta and Wayne Burleson. Low-power, process-variation tolerant on-chip thermal monitoring using track and hold based thermal sensors. In *Proceedings of the 19th ACM Great Lakes symposium on VLSI*, pages 145–148, New York, New York, USA, 2009. ACM.

[17] RH Dennard, FH Gaensslen, HN Yu, VL Rideout, E Bassous, and AR LeBlanc. Design of ion-implanted MOSFET's with very small physical dimensions. *Proceedings of the IEEE Journal of Solid-State Circuits*, pages 256–268, 1974.

[18] M.K. Gowan, L.L. Biro, and D.B. Jackson. Power considerations in the design of the Alpha 21264 microprocessor. In *Design Automation Conference, 1998. Proceedings*, pages 726–731, 1998.

[19] P.E. Gronowski, W.J. Bowhill, R.P. Preston, M.K. Gowan, and R.L. Allmon. High-performance microprocessor design. *IEEE Journal of Solid-State Circuits*, 33(5):676–686, 1998.

[20] Yongkui Han. *Temperature aware techniques for design, simulation and measurement in microprocessors*. PHD, University of Massachusetts Amherst, 2007.

[21] W Huang, K Sankaranarayanan, and RJ Ribando. An improved block-based thermal model in HotSpot 4.0 with granularity considerations. *Proceedings of the Workshop on Duplicating*, 2007.

[22] W Huang, MR Stan, K Sankaranarayanan, RJ Ribando, and K. Skadron. Many-Core Design from a Thermal Perspective: Extended Analysis and Results. *Work*, 5(June):1–12, 2008.

[23] Wei Huang, Kevin Skadron, Sudhanva Gurumurthi, Robert J. Ribando, and Mircea R. Stan. Exploring the thermal impact on manycore processor performance. *2010 26th Annual IEEE Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM)*, pages 191–197, February 2010.

[24] Wei Huang, Mircea R. Stan, Sudhanva Gurumurthi, Robert J. Ribando, and Kevin Skadron. Interaction of scaling trends in processor architecture and cooling. *2010 26th Annual IEEE Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM)*, pages 198–204, February 2010.

[25] Wei Huang, Mircea R. Stant, Karthik Sankaranarayanan, Robert J. Ribando, and Kevin Skadron. Many-core design from a thermal perspective. *Proceedings of the 45th Annual Conference on Design Automation*, page 746, 2008.

[26] C. Hung, W.Addo-Quaye, T. Theocharides, Y. Xie, N. Vijakrishnan, and M.J. Irwin. Thermal-aware IP virtualization and placement for networks-on-chip architecture. In *IEEE International Conference on Computer Design: VLSI in Computers and Processors. ACM Press*, pages 430–437, 2004.

[27] Intel. Mobile Intel Pentium 4 Processor-M, 2003.

[28] Intel. Intel Core 2 Duo Mobile Processor for Intel Centrino Duo Mobile Processor Technology, 2007.

[29] Stefanos Kaxiras and P Xekalakis. 4T-Decay sensors: a new class of small, fast, robust, and low-power, temperature/leakage sensors. *International Symposium on Low Power Electronics and Design*, pages 108–113, 2004.

[30] R.E. Kessler, E.J. McLellan, and D.A. Webb. The Alpha 21264 Microprocessor Architecture, 1998.

[31] Peter Kogge, K. Bergman, S. Borkar, and D. Campbell. Exascale computing study: Technology challenges in achieving exascale systems. 2008.

[32] Peter M Kogge. Exascale Computing: Embedded Style. *HPEC 2009 Proceedings*, 2009.

[33] Vitaly Krinitsin. Pentium 4 and Athlon XP Thermal Conditions.

[34] R. Kumar and V. Kursun. Impact of temperature fluctuations on circuit characteristics in 180nm and 65nm CMOS technologies. *2006 IEEE International Symposium on Circuits and Systems*, page 4, 2006.

[35] R. Kumar, D.M. Tullsen, N.P. Jouppi, and P. Ranganathan. Heterogeneous chip multiprocessors. *Computer*, 38(11):32–38, November 2005.

[36] R. Kumar, V. Zyuban, and D.M. Tullsen. Interconnections in Multi-Core Architectures: Understanding Mechanisms, Overheads and Scaling. *32nd International Symposium on Computer Architecture (ISCA'05)*, pages 408–419, 2005.

[37] K.J. Lee and K. Skadron. Using performance counters for runtime temperature sensing in high-performance processors. *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 11*, 2005.

[38] Frank Y S Lin and P L Chiu. A Near-Optimal Sensor Placement Algorithm to Achieve Complete Coverage/Discrimination in Sensor Networks. *IEEE Communications Letters*, 9(1), 2005.

[39] Frank Liu. A General Framework for Spatial Correlation Modeling in VLSI Design. *2007 44th ACM/IEEE Design Automation Conference*, pages 817–822, June 2007.

[40] Yongpan Liu, Robert P. Dick, Li Shang, and Huazhong Yang. Accurate Temperature-Dependent Integrated Circuit Leakage Power Estimation is Easy. *2007 Design, Automation & Test in Europe Conference & Exhibition*, pages 1–6, April 2007.

[41] J. Long, S.O. Memik, G. Memik, and R. Mukherjee. Thermal monitoring mechanisms for chip multiprocessors. *ACM Transactions on Architecture and Code Optimization*, 5(2):1–33, 2008.

[42] J Macqueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[43] W. McMahon, A. Haggag, and K. Hess. Reliability scaling issues for nanoscale devices. *IEEE Transactions On Nanotechnology*, 2(1):3338, March 2003.

[44] Seda Ogrenci Memik, Rajarshi Mukherjee, Min Ni, and Jieyi Long. Optimizing Thermal Sensor Allocation for Microprocessors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(3):516–527, March 2008.

[45] R. Mukherjee and S.O. Memik. Systematic temperature sensor allocation and placement for microprocessors. In *Proceedings of the 43rd Annual Design Automation Conference*, page 547. ACM, 2006.

[46] Rajarshi Mukherjee, Somsubhra Mondal, and Seda Memik. Thermal Sensor Allocation and Placement for Reconfigurable Systems. *2006 IEEE/ACM International Conference on Computer Aided Design*, pages 437–442, November 2006.

[47] S.S. Mukherjee, P. Bannon, S. Lang, a. Spink, and D. Webb. The Alpha 21364 network architecture. *IEEE Micro*, 22(1):26–35, 2002.

[48] A.N. Nowroz, Ryan Cochran, and S. Reda. Thermal Monitoring of Real Processors: Techniques for Sensor Allocation and Full Characterization. *Order A Journal On The Theory Of Ordered Sets And Its Applications*, pages 56–61, 2010.

[49] M. Pedram and S. Nazarian. Thermal Modeling, Analysis, and Management in VLSI Circuits: Principles and Methods. *Proceedings of the IEEE*, 94(8):1487–1501, August 2006.

[50] D Pham, S Asano, M Bollinger, and M.N. Day. The design and implementation of a first generation cell processor. *ISSCC Microprocessors and Signal Processing*, 10(2):184–186, 2005.

[51] F Pollack. New microarchitecture challenges in the coming generations of cmos process technologies. *Keynote speech: 32nd International Symposium on Microarchitecture*, pages 1–34, 1999.

[52] E. Rotem, A. Naveh, M. Moffie, and A. Mendelson. Analysis of thermal monitor features of the intel pentium m processor. In *TACS Workshop at ISCA-31*, pages 29–35. 2004.

[53] Efraim Rotem, J. Hermerding, A. Cohen, and H. Cain. Temperature measurement in the Intel Core Duo Processor. *Legacy*, pages 1–5, 2006.

[54] Mert R. Sabuncu and Peter J. Ramadge. Gradient based nonuniform subsampling for information-theoretic alignment methods. *Proceedings of the 26th Annual International Conference of the IEEE EMBS*, pages 1683–1686, 2004.

[55] H. Sanchez, B. Kuttanna, T. Olson, M. Alexander, G. Gerosa, R. Philip, and J. Alvarez. Thermal management system for high performance PowerPC microprocessors. *Proceedings IEEE COMPCON 97. Digest of Papers*, pages 325–330, 1997.

[56] K Sankaranarayanan. *Thermal Modeling and Management of Microprocessors*. Phd, Univ. of Virginia School of Engineering and Applied Science, 2009.

[57] K Sankaranarayanan, S Velusamy, M. Stan, and K. Skadron. A case for thermal-aware floorplanning at the microarchitectural level. *Journal of Instruction-Level Parallelism*, 7:1–16, 2005.

[58] Greg Semeraro, Grigorios Magklis, Rajeev Balasubramonian, David H. Albonesi, Sandhya Dwarkadas, and Michael L Scott. Energy-Efficient Processor Design Using Multiple Clock Domains with Dynamic Voltage and Frequency Scaling. *Proceedings of the 8th International Symposium on High-Performance Computer Architecture*, 2002.

[59] Semiconductor Industries Association. International Technology Roadmap, 2009.

[60] Shervin Sharifi, C.C. Liu, and T.S. Rosing. Accurate temperature estimation for efficient thermal management. In *9th International Symposium on Quality Electronic Design*, pages 137–142. 2008.

[61] T. Sherwood, E. Perelman, and B. Calder. Basic block distribution analysis to find periodic behavior and simulation points in applications. *Proceedings 2001 International Conference on Parallel Architectures and Compilation Techniques*, pages 3–14, 2001.

[62] K. Skadron and W. Huang. Analytical model for sensor placement on microprocessors. *2005 International Conference on Computer Design*, pages 24–27, 2005.

[63] K. Skadron and K.J. Lee. Using Performance Counters for Runtime Temperature Sensing in High-Performance Processors. *19th IEEE International Parallel and Distributed Processing Symposium*, page 8. 2005.

[64] K Skadron, MR Stan, W Huang, and S Velusamy. Temperature-aware microarchitecture: Extended discussion and results. *University of Virginia, Department of Computer Science*, 2003.

[65] K. Skadron, M.R. Stan, W. Huang, S. Veluswamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture: Modeling and implementation. *ACM Transactions on Architecture and Code*, 2004.

[66] SPEC-CPU2000. Standard Performance Evaluation Council, Performance Evaluation in the New Millennium, Version 1.1, 2000.

[67] J. Srinivasan, S.V. Adve, P. Bose, and J.a. Rivers. The case for lifetime reliability-aware microprocessors. *Proceedings. 31st Annual International Symposium on Computer Architecture, 2004.*, 32(2):276–287, 2004.

[68] V. Szekely, M. Rencz, and B. Courtois. Tracing the thermal behavior of ICs. *IEEE Design & Test of Computers*, 15(2):14–21, 1998.

[69] Xiang Yun. *On-Chip Thermal Sensor Placement*. Master's Thesis, University of Massachusetts Amherst, 2008.

[70] Jia Zhao, Sailaja Madduri, Ramakrishna Vadlamani, Wayne Burleson, and Russell Tessier. A Dedicated monitoring infrastructure for multicore processors. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2010.